A Domain Decomposition Approach to Finite Volume Solutions of the Euler Equations on Triangular Meshes

V. Dolean, Stephane Lanteri

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

A domain decomposition approach to finite volume solutions of the Euler equations on triangular meshes

V. Dolean and S. Lanteri

N° 3751

Août 1999

THÈME 4

ISSN 0249-6399

INRIA
SOPHIA ANTIPOLIS

A domain decomposition approach to finite volume solutions of the Euler equations on triangular meshes

V. Dolean* and S. Lanteri*

Thème 4 — Simulation et optimisation
de systèmes complexes
Projet Sinus

Rapport de recherche n° 3751 — Août 1999 — 59 pages

Abstract: we report on our recent efforts on the formulation and the evaluation of a domain decomposition algorithm for the parallel solution of two-dimensional compressible inviscid flows. The starting point is a flow solver for the Euler equations which is based on a combined finite element/finite volume formulation on unstructured triangular meshes for the spatial discretization. Time integration of the resulting semi-discrete equations is obtained using a linearized backward Euler implicit scheme. As a result, each pseudo time step requires the solution of a sparse linear system for the flow variables. In this study, a non-overlapping domain decomposition algorithm is used for advancing the solution at each implicit time step. First, we formulate an additive Schwarz algorithm using appropriate matching conditions at the subdomain interfaces. In accordance with the hyperbolic nature of the Euler equations, these transmission conditions are Dirichlet conditions for the characteristic variables corresponding to incoming waves. Then, we introduce interface operators that allow to express the domain decomposition algorithm as a Richardson type iteration on the interface unknowns. Algebraically speaking, the Schwarz algorithm is equivalent to a Jacobi iteration applied to a linear system whose matrix has a block structure. A substructuring technique can be applied to this matrix in order to obtain a fully implicit scheme in terms of interface unknowns. In our approach, the interface unknowns are numerical (normal) fluxes.

Key-words: Domain decomposition method - Euler Equations - Finite volumes - Triangular meshes - Multigrid algorithm - Parallel computing

* INRIA, 2004 Route des Lucioles, BP. 93, 06902 Sophia Antipolis Cédex-France

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles, B.P. 93, 06902 Sophia Antipolis Cedex (France)
Téléphone : 04 92 38 77 77 - International : +33 4 92 38 77 77 — Fax : 04 92 38 77 65 - International : +33 4 92 38 77 65

Une approche par décomposition de domaine pour la résolution des équations d'Euler par une méthode volumes finis en maillages triangulaires

Résumé : dans ce rapport, on formule et on évalue numériquement un algorithme par décomposition de domaine pour la résolution parallèle d'écoulements bidimensionnels de fluides parfaits. Le point de départ de notre étude est constitué d'un solveur des équations d'Euler qui repose sur une formulation mixte éléments finis/volumes finis en maillages triangulaires pour la discrétisation en espace. L'intégration en temps est réalisée au moyen d'une méthode d'Euler implicite linéarisée. Chaque pas de temps nécessite la résolution approchée d'un système linéaire de matrice creuse non-symétrique. Ici, on utilise un algorithme par décomposition de domaine non-recouvrant pour la réalisation d'un pas de temps implicite. On commence par formuler un algorithme de type Schwarz additif basé sur des conditions d'interface prenant en compte la nature hyperbolique des équations d'Euler. Plus précisément, ces conditions sont de type Dirichlet et portent sur les variables caractéristiques associées aux ondes entrantes dans un sous-domaine. On introduit ensuite des opérateurs interface permettant de formuler l'algorithme comme une itération de type Richardson sur un ensemble d'inconnues définies aux interfaces. D'un point de vue algébrique, l'algorithme de Schwarz peut être interprété comme une méthode de relaxation de Jacobi appliquée à la résolution d'un système linéaire dont la matrice a une structure par blocs particulière. Une technique de sous-structuration peut être appliquée à cette matrice afin d'obtenir un système interface. Dans notre cas, les variables aux interfaces sont les composantes de flux numériques décentrés. Il en résulte un algorithme par décomposition de domaine du type complément de Schur.

Mots-clés : Méthode de décomposition de domaine - Equations d'Euler - Volumes finis - Maillages triangulaires - Algorithme multigrille - Calcul parallèle

1 Introduction

When solving a problem modelled by a partial differential equation, one is generally confronted to a discretization step followed by a linear system solve. The size or the ill-conditioning of the latter makes a global or a direct solution approach quite inappropriate. With the advent of parallel computers, domain decomposition algorithms have enjoyed an increasing popularity among the scientific community because they define a good framework to derive efficient solvers for the resulting systems using the mathematical properties of the initial PDE[38]. Indeed, since the early 1980's, efficient and scalable domain decomposition algorithms have been developed for the solution of computational structural mechanics problems (i.e. for the solution of elliptic PDEs); however their application to computational fluid dynamics problems (i.e. to the solution of hyperbolic or mixed hyperbolic/parabolic PDEs) has been less remarkable.

In [38] the term « domain decomposition » has a few different meanings: data distribution between the processors of a distributed memory computer, splitting of a physical domain in regions where different physical models can be used, or, from a preconditioning point of view, splitting of the global problem in smaller ones thar are easily solved to provide an approximate inverse to the original operator. As a general rule, domain decomposition algorithms allow to solve problems of large size by decomposing them in smaller size ones that can be treated by several computers with low memory capacity. In this case the solution of a subproblem provides the definition of the boundary conditions for the next subproblem when proceeding with a sequential treatement of the different subproblems. The Schwarz algorithm which illustrates this approach relies on a spatial decomposition of the initial domain in *overlapping* subdomains. Its original form is often called *multiplicative* since the corresponding iterative operator can be expressed as the product of some local operators related to the local resolutions. It is obvious that this kind of algorithm is not well suited to parallel architectures. Lately, *additive* variants have been devised that are characterized by a simultaneous treatement of the subproblems and therefore efficiently exploit parallel computing architectures. To summarize, domain decomposition methods can be classified according to two criteria : *overlapping* versus *non-overlapping* methods according to the spatial decomposition of the global domain, *multiplicative* versus *additive* ones according to the interdependence of the local solutions at each iteration. The non-overlapping domain decomposition methods can be of the Schwarz or substructuring (Schur complement or interface system) types, the latest being related to block Gaussian elimination techniques (each block corresponding to a different subdomain). When solving an interface problem, one deals with an operator acting on interface variables, whose discretisation is the Schur complement of the global operator[34],[32].

Domain decomposition methods were first developed for elliptic second-order problems, taking advantage of the strong regularity of their solutions as well as of the symmetry of the operators involved (or the dominance of the symmetric part of the operators)[42],[3],[37]. The situation is less clear for hyperbolic or mixed hyperbolic parabolic models of compressible fluid mechanics. One has to deal with first-order PDE characterized by non-symmetric operators, with possible singular solutions. When the symmetric part is dominant one can still apply the algorithms built for the symmetric systems with a few modifications. If not, for example when convection is dominant in the convection-diffusion case, different approaches exist using Dirichlet and/or Neumann interface conditions as in [8], or using a Robin transmission condition and an *iteration by subdomain* algorithm as in [4], [14] and [15].

In order to accelerate the convergence of non-overlapping domain decomposition algorithms for the solution of convection-diffusion problems, one can basically consider two directions: construct an appropriate preconditioning method for the resulting interface problem, or modify the interface conditions involved in a Schwarz type algorithm. For instance, in [1] an optimal Robin-Robin preconditioner is built from local problems with Robin type conditions at the interface. On the other hand, the second acceleration strategy relies on the notion of absorbing boundary conditions. The absorbing boundary conditions (or artificial boundary conditions) have been introduced for the first time by Engquist and Majda[9] for the solution of PDE on an unbounded domain. They are imposed on an artificial boundary such that the solution on the truncated domain is the restriction of the whole solution. Their application to the convection-diffusion or Navier-Stokes equations has been studied by Halpern and Schatzmann in [18], [19] and [20]. Recently, Nataf[28] discovered a similar situation in domain decomposition. Nataf *et al.*[30] showed that in this context, the use of the absorbing boundary conditions leads to an optimal convergence of a Schwarz type algorithm. However, this type of boundary conditions involves the use of non local operators that need be approximated by partial differential operators. This has been done in the case of a convection-diffusion equation by Japhet[22] using, as an approximation criterion, the minimization of the convergence rate of a Schwarz type algorithm whose transmission conditions have been optimized.

The objective of the present work is to solve the Euler equations for compressible flows by a non overlapping domain decomposition method, and more precisely by a substructuring method. The formulation of a Schwarz type algorithm for the Euler equations can be found in [6] and in [31]. In [13] one can find an interface formulation for scalar transport equations by defining a Steklov-Poincaré type operator. Clerc[6]

considered different classes of transmission conditions applied to the solution of linearized and symmetrized hyperbolic systems such as the Cauchy-Riemann equations. Here, we study « classical » transmission conditions that are derived naturally from a weak formulation of the Euler equations. The starting point is a flow solver for the Euler equations which is based on a combined finite element/finite volume formulation on unstructured triangular meshes for the spatial discretization. Time integration of the resulting semi-discrete equations is obtained using a linearized backward Euler implicit scheme[12]. As a result, each pseudo time step requires the solution of a sparse linear system for the flow variables. In this work, a non-overlapping domain decomposition algorithm is used for advancing the solution at each implicit time step. First, we formulate an additive Schwarz algorithm using appropriate matching conditions at the subdomain interfaces. In accordance with the hyperbolic nature of the Euler equations, these transmission conditions are Dirichlet conditions for the characteristic variables corresponding to incoming waves [31]. Then, we introduce interface operators that allow to express the domain decomposition algorithm as a Richardson type iteration on the interface unknowns. Algebraically speaking, the Schwarz algorithm is equivalent to a Jacobi iteration applied to a linear system whose matrix has a block structure. A substructuring technique can be applied to this matrix in order to obtain a fully implicit scheme in terms of interface unknowns. In our approach, the interface unknowns are numerical (normal) fluxes.

The remaining part of the paper is organised as follows. In section 2, the Schwarz algorithm and the substructuring technique are introduced in the continuous case for a general linear hyperbolic system. Section 3 describes the characteristics of the starting point mixed finite element/finite volume flow solver for the Euler equations. The proposed domain decomposition approach is then adapted to the discrete case in section 4. For steady flow calculations, the linear system resulting from the implicit scheme is generally solved approximately (for example, in the original solver, approximate solutions are obtained using relaxation methods such as the Jacobi or Gauss-Seidel methods). Here, we have adopted the same approach for the local solutions induced by the domain decomposition algorithm. In particular, we do not perform direct solution of local problems. In this paper, this strategy is only justified through numerical experiments as we compare the convergence of the proposed domain decomposition algorithm for completely converged and approximate solutions of the local problems. In order to improve the overall efficiency of the domain decomposed flow solver, the iterative solution of local linear systems based on Jacobi or Gauss-Seidel relaxation methods is accelerated by a linear multigrid strategy by volume agglomeration[23]. This is described in section 5. In section 6, the resulting domain decomposed flow solver is evaluated through numerical experiments that are performed on a cluster

of `Pentium Pro` computers interconnected via a 100 Mbit/s `FastEthernet` switch. Finally, conclusions and future works are presented in section 7.

Before proceeding with the next section, we precise that theoretical aspects related to the present work such as the convergence analysis of the Schwarz algorithm, the construction of optimal preconditioners for the interface system and the justification of the approximate local solution strategy are the object of ongoing works that will be related in separate papers.

2 Domain decomposition for hyperbolic systems

In this section we outline the basic principles for the formulation of non-overlapping domain decompositon algorithms for hyperbolic systems of partial differential equations. The proposed framework is greatly inspired from Gastaldi and Gastaldi[13], Gastaldi *et al.*[14], Quarteroni and Stolcis[31], Quarteroni and Valli[32], Quarteroni and Valli[33] and Nataf[28]. We also refer to Desideri[7], Smith *et al.*[38] and Quarteroni and Valli[34] for a general introduction, as well as an in-depth description of domain decomposition algorithms. Most of the discussion here is undertaken in the context of a general linear hyperbolic system. Then, in the next section, we naturally extend the proposed ideas to the solution of the Euler equations for compressible flows.

2.1 Hyperbolic systems and boundary conditions

In this paper we are interested in the numerical solution of a system of conservation laws of the form :

$$\partial_t W + \sum_{i=1}^{d} \partial_{x_i} F_i(W) = 0, \quad W \in \mathbb{R}^p \tag{1}$$

where d denotes the space dimension and p the dimension of the system. The flux functions F_i are assumed differentiable with respect to the state vector $W = W(x,t)$. In the general case, the flux functions are non-linear functions of W. In addition, if W is assumed regular, system (1) can be written in quasi-linear form :

$$\partial_t W + \sum_{i=1}^{d} \frac{\partial F_i}{\partial W}(W) \partial_{x_i} W = 0 \tag{2}$$

or :

$$\partial_t W + \sum_{i=1}^{d} A_i(W)\partial_{x_i} W = 0 \tag{3}$$

The $A_i(W) = \dfrac{\partial F_i}{\partial W}(W)$ are the Jacobian matrices of the flux functions $F_i(W)$, with respect to W. Recall that system (1) is said to be hyperbolic if, for any unitary real vector $\boldsymbol{n} \in \mathbb{R}^d$, the matrix $\displaystyle\sum_{i=1}^{d} A_i(W)n_i$ is diagonalizable with real eingenvalues.

We are particularly interested in the situation where system (1) is integrated in time using a backward Euler implicit scheme involving a linearization of the flux functions. In that case we have :

$$\frac{\delta W}{\delta t} + \sum_{i=1}^{d} \partial_{x_i} \left[\frac{\partial F_i}{\partial W}(W^n)\delta W \right] = -\mathrm{div}(F(W^n)) \tag{4}$$

where $\delta W = W(x, t^{n+1}) - W(x, t^n) = W^{n+1} - W^n$. When δW is assumed regular, we can write the non-conservative form of system (4) :

$$\left[\frac{1}{\delta t} Id + \sum_{i=1}^{d} \partial_{x_i} \left[\frac{\partial F_i}{\partial W}(W^n) \right] \right] \delta W + \sum_{i=1}^{d} \left[\frac{\partial F_i}{\partial W}(W^n) \right] \partial_{x_i} \delta W = -\mathrm{div}(F(W^n)) \tag{5}$$

System (5) can be symmetrized through the multiplication of an operator Σ (see for example Barth[2]) which, for hyperbolic systems admitting an entropy function, is given by the Hessian matrix of this entropy. This operation results in the following first order system :

$$A_0 \delta W + \sum_{i=1}^{d} A_i \partial_{x_i} \delta W = f \tag{6}$$

with :

$$\begin{cases} A_0 &= \Sigma \left[\dfrac{1}{\delta t} Id + \displaystyle\sum_{i=1}^{d} \partial_{x_i} \left[\frac{\partial F_i}{\partial W}(W^n) \right] \right] \\ A_i &= \Sigma \left[\dfrac{\partial F_i}{\partial W}(W^n) \right] \\ f &= -\Sigma \mathrm{div}(F(W^n)) \end{cases} \tag{7}$$

Now, let $\boldsymbol{n} = (n_1, ..., n_p)$ denote the outward normal vector to $\partial\Omega$; we define $A_{\boldsymbol{n}}W$ as the normal trace of W on $\partial\Omega$, with $A_{\boldsymbol{n}} = \sum_{i=1}^{d} A_i n_i$. When dealing with boundary conditions, it is well known that one cannot impose all the components of W on the boundary $\partial\Omega$. Instead, the direction of propagation of the information has to be taken into account in order to obtain a well posed initial and boundary value problem (IBVP) for system (6). More precisely, the number and type of boundary conditions that must be imposed on $\partial\Omega$ are deduced from the expression of system (6) in terms of characteristic variables and is related to information entering the domain Ω. A more rigourous discussion of boundary conditions treatment for hyperbolic systems from gas dynamics, in terms of characteristic variables, is for example given in [16] (see also Quarteroni and Valli[33] for a discussion in the context of domain decomposition algorithms). In order to do so, we decompose the operator $A_{\boldsymbol{n}}$ in positive and negative parts i.e. $A_{\boldsymbol{n}} = A_{\boldsymbol{n}}^{+} + A_{\boldsymbol{n}}^{-}$. Using the diagonalization of $A_{\boldsymbol{n}}$ which writes as $A_{\boldsymbol{n}} = T\Lambda_{\boldsymbol{n}}T^{-1}$ we have :

$$\left\{ \begin{array}{c} A_{\boldsymbol{n}}^{\pm} = T\Lambda_{\boldsymbol{n}}^{\pm}T^{-1} \\ \Lambda_{\boldsymbol{n}}^{\pm} = \mathrm{diag}(\lambda_i^{\pm})_{1\leq i\leq p} \quad \text{with} \quad \lambda_i^{\pm} = \dfrac{1}{2}(\lambda_i \pm |\lambda_i|) \\ \text{and with} \quad A_{\boldsymbol{n}}^{+}W = -A_{-\boldsymbol{n}}^{-}W \end{array} \right.$$

In order to obtain a well posed IBVP, we have to impose boundary conditions of the form :

$$A_{\boldsymbol{n}}^{-}W = A_{\boldsymbol{n}}^{-}g \tag{8}$$

where $A_{\boldsymbol{n}}^{-}$ is used to select the information entering the domain Ω. Then, a well known result is that the problem :

$$\left\{ \begin{array}{rcl} \mathcal{L}W = A_0W + \sum_{i=1}^{d} A_i\partial_{x_i}W &=& f, \quad \text{in } \Omega \\ A_{\boldsymbol{n}}^{-}W &=& A_{\boldsymbol{n}}^{-}g, \quad \text{on } \partial\Omega \end{array} \right. \tag{9}$$

with $f \in L^2(\Omega)^p$ and $g \in L_A^2(\partial\Omega)$, has a unique solution $W \in \tilde{H}$ (see for example [6]) with :

$$\tilde{H} = \{W \in L^2(\Omega)^p \text{ such that } \sum_{i=1}^{d} A_i\partial_{x_i}W \in L^2(\Omega)^p \text{ and } W|_{\partial\Omega} \in L_A^{1/2}(\partial\Omega)\}$$
$$\text{with } L_A^{1/2}(\partial\Omega) = \{W \text{ such that } \int_{\partial\Omega} |A_{\boldsymbol{n}}|W \cdot W d\sigma < \infty\}$$

This result is used in the next section to formulate a non-overlapping domain decomposition algorithm for the solution of (9).

2.2 Domain decomposition and interface conditions

The domain decomposition approach for solving (9) consists in defining well posed subproblems so that a local solution on a given subdomain Ω_i is the restriction of the global solution on Ω to Ω_i. The sub-problems will inherit the physical boundary conditions of the global problem for the part of $\partial\Omega_i$ which intersects $\partial\Omega$; in addition, appropriate interface conditions have to be added to the definition of the subproblems for the part of $\partial\Omega_i$ which is common to neighboring subdomains. We shall introduce a non-overlapping domain decomposition algorithm for the following boundary value problem :

$$\begin{cases} \mathcal{L}W \equiv A_0 W + \sum_{k=1}^{d} A_k \partial_{x_k} W = f \ \text{ in } \ \Omega \\ \text{Boundary conditions on } \ \partial\Omega \end{cases} \tag{10}$$

Let $\Omega = \bigcup_{i=1}^{N} \Omega_i$ be a stripwise (for simplicity of presentation) decomposition of Ω and W_i the solution of the local problem :

$$\begin{cases} \mathcal{L}W_i = f|_{\Omega_i} = f_i \\ \text{Boundary conditions on } \ \partial\Omega \cap \partial\Omega_i \\ \text{Interface conditions on } \ \partial\Omega_i \cap \partial\Omega_j \end{cases} \tag{11}$$

Let $\boldsymbol{n_i}$ be the outward normal to $\partial\Omega_i$. The local solution W_i is prolongated by zero on Ω/Ω_i; then a necessary and sufficient condition to insure that $\sum_{i=1}^{N} W_i$ is the solution of the global problem (10) is that the interface conditions on $\Gamma = \partial\Omega_i \cap \partial\Omega_j$ take the form :

$$\begin{cases} A^-_{\boldsymbol{n_i}} W_i + A^+_{\boldsymbol{n_j}} W_j = 0 \\ \text{and} \\ A^+_{\boldsymbol{n_i}} W_i + A^-_{\boldsymbol{n_j}} W_j = 0 \end{cases} \tag{12}$$

We are going now to demonstrate this result in the two-subdomain case $\Omega = \Omega_1 \cup \Omega_2$. Let $\boldsymbol{n}_{\partial\Omega}$ denote the outward normal on $\partial\Omega$ and $\boldsymbol{n} = \boldsymbol{n}_1 = -\boldsymbol{n}_2$ the outward normal

on Γ (directed from Ω_1 to Ω_2), and let $W = W_1 + W_2$ and $f = f_1 + f_2$. We introduce variational formulations of (10)-(11) which are based on the following space of test functions :

$$V = \{X \in L^2(\Omega)^p \text{ such that } \quad \sum_{i=1}^{d} A_i X \in H(\mathrm{div}, \Omega)^p, \quad A_{\boldsymbol{n}} X \in L_A^{1/2}(\Gamma)\}$$

On one hand, we integrate by parts on the global domain Ω and, on the other hand, we integrate by parts on each subdomain Ω_1 and Ω_2 :

$$\begin{aligned}
\int_\Omega [A_0 W + \sum_{k=1}^{d} A_k \partial_{x_k} W] X &= \int_\Omega [A_0^T X - \sum_{k=1}^{d} \partial_{x_k}(A_k X)] W + \int_{\partial\Omega} (A_{\boldsymbol{n}_{\partial\Omega}} W) X \\
&= \int_{\Omega_1} [A_0^T X - \sum_{k=1}^{d} \partial_{x_k}(A_k X)] W_1 + \int_{\partial\Omega_1 \cap \partial\Omega} (A_{\boldsymbol{n}_{\partial\Omega}} W_1) X \\
&+ \int_{\Omega_2} [A_0^T X - \sum_{k=1}^{d} \partial_{x_k}(A_k X)] W_2 + \int_{\partial\Omega_2 \cap \partial\Omega} (A_{\boldsymbol{n}_{\partial\Omega}} W_2) X \\
&= \int_{\Omega_1} [A_0 W_1 + \sum_{k=1}^{d} A_k \partial_{x_k} W_1] X - \int_\Gamma (A_{\boldsymbol{n}_1} W_1) X \\
&+ \int_{\Omega_2} [A_0 W_2 + \sum_{k=1}^{d} A_k \partial_{x_k} W_2] X - \int_\Gamma (A_{\boldsymbol{n}_2} W_2) X \\
&= \int_{\Omega_1} [A_0 W_1 + \sum_{k=1}^{d} A_k \partial_{x_k} W_1] X \\
&+ \int_{\Omega_2} [A_0 W_2 + \sum_{k=1}^{d} A_k \partial_{x_k} W_2] X \\
&+ \int_\Gamma [A_{\boldsymbol{n}} W_2 - A_{\boldsymbol{n}} W_1] X
\end{aligned}$$

If W_1 and W_2 are the (local) solutions of (11) then we must have :

$$\int_\Gamma [A_{\boldsymbol{n}} W_1 - A_{\boldsymbol{n}} W_2] X = 0 \quad \forall X \in V$$

Therefore a necessary and sufficient condition to insure that $W_1 + W_2$ is the (global) solution of (10) is :

$$A_{\boldsymbol{n}} W_1 = A_{\boldsymbol{n}} W_2 \quad \text{on} \quad \Gamma \tag{13}$$

Since $A_{\boldsymbol{n}}$ is non-singular then (13) implies that :

$$W_1 = W_2 \quad \text{on} \quad \Gamma \tag{14}$$

therefore :

$$\left(T\Lambda_{\boldsymbol{n}}^{\pm}T^{-1}\right) W_1 = \left(T\Lambda_{\boldsymbol{n}}^{\pm}T^{-1}\right) W_2 \quad \text{on} \quad \Gamma \tag{15}$$

which yields :

$$\left\{ \begin{array}{ccc} A_{\boldsymbol{n}}^{-}W_1 & = & A_{\boldsymbol{n}}^{-}W_2 \\ & \text{and} & \\ A_{\boldsymbol{n}}^{+}W_1 & = & A_{\boldsymbol{n}}^{+}W_2 \end{array} \right. \tag{16}$$

Conversely, since $A_{\boldsymbol{n}} = A_{\boldsymbol{n}}^{+} + A_{\boldsymbol{n}}^{-}$, conditions (16) imply (13), which concludes the proof.

2.3 A non-overlapping additive Schwarz algorithm

For simplicity of presentation, we assume that the domain Ω is rectangular $\Omega = [x_a, x_b] \times [y_a, y_b]$ and we consider the case of a non-overlapping decomposition in vertical strips where the subdomains are defined by $\Omega_i =]\gamma_{i-1}, \gamma_i[\times]y_a, y_b[,\ 2 \leq i \leq N-1,\ \ \Omega_1 =]x_a, \gamma_1[\times]y_a, y_b[,\quad \Omega_N =]\gamma_{N-1}, x_b[\times]y_a, y_b[$ (see Fig. 3). So the outward normal vectors at the interfaces for the subdomain Ω_i are $\boldsymbol{n}_{i,l} = (-1, 0)$ and $\boldsymbol{n}_{i,r} = (1, 0)$. Consequently, $A_{\boldsymbol{n}_{i,l}}^{-} = -A_{\boldsymbol{n}_{i-1,r}}^{+} = -A^{+}$ and $A_{\boldsymbol{n}_{i-1,r}}^{-} = A^{-}$. We further define a Schwarz type algorithm where the interface transmission conditions are well adapted to the hyperbolic nature of the problem as in [6]. Let $W_i^{(0)}$ denote the initial appoximation of the solution in subdomain Ω_i, then the approximation at the $(k+1)$-th iteration (where k defines the iteration of the Schwarz algorithm) is the solution of the problem :

$$\left\{ \begin{array}{lcll} \mathcal{L}W_i^{(k+1)} & = & f & \text{in } \Omega_i \\ A^{+}W_i^{(k+1)} & = & A^{+}W_{i-1}^{(k)} & \text{on } \Gamma_{i,l} \\ A^{-}W_i^{(k+1)} & = & A^{-}W_{i+1}^{(k)} & \text{on } \Gamma_{i,r} \\ A_{\boldsymbol{n}}^{-}W_i^{(k+1)} & = & A_{\boldsymbol{n}}^{-}g & \text{on } \partial\Omega \cup \partial\Omega_i \end{array} \right. \tag{17}$$

with the convention that $\Gamma_{i,l}$ (respectively $\Gamma_{i,r}$) is the straight line $x = \gamma_{i-1}$ (respectively $x = \gamma_i$). Moreover, we have that $W_i^{(0)} = W_i^n$ and $W_i^{n+1} = W_i^{(K)}$, K being the number

of iterations of the above algorithm (n denotes the time step). Clearly, (17) defines an additive Schwarz type algorithm even though its formulation is somewhat unconventional as it is based on a non-overlapping partitioning of the domain Ω. Such algorithms have been extensively studied by Nataf[28] and Nataf *et al.*[30] for convection-diffusion problems. In particular, these authors have considered the use of high-order optimal interface conditions, inspired from the concept of absorbing boundary conditions for unbounded domains[9], for improving the convergence of the Schwarz algorithm.

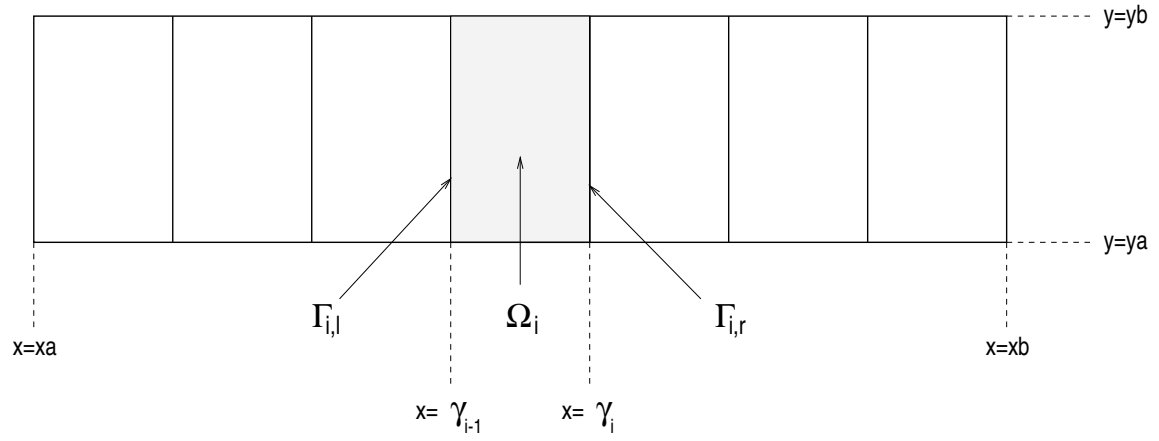


Figure 1: Definition of vertical strips decomposition

2.4 Substructuring for the definition of an interface problem

In this section we reformulate the previous additive Schwarz algorithm as the iterative solution of an interface system stated in terms of normal fluxes at subdomain interfaces. Adopting the formalism used in [30], we introduce interface operators for each subdomain, that take as argument the incoming fluxes at the two interfaces and return outgoing fluxes :

$$S^i : L_A^{1/2}(\Gamma_{i,l})^p \times L_A^{1/2}(\Gamma_{i,r})^p \times L^2(\Omega_i)^p \times L^2(\Omega_i)^p \quad \longrightarrow \quad L_A^{-1/2}(\Gamma_{i,l})^p \times L_A^{-1/2}(\Gamma_{i,r})^p$$

such that :

$$S^i \ : \ (\Phi_{i,l} \ , \ \Phi_{i,r} \ , \ f \ , \ g) \quad \longrightarrow \quad \left(A^- W_i|_{\Gamma_{i,l}} \ , \ A^+ W_i|_{\Gamma_{i,r}}\right) \tag{18}$$

for $2 \leq i \leq N-1$, and :

$$\begin{array}{rcl} S^1 & : & L_A^{1/2}(\Gamma_{1,r})^p \times L^2(\Omega_i)^p \times L^2(\Omega_i)^p \longrightarrow L_A^{-1/2}(\Gamma_{1,r})^p \\ S^N & : & L_A^{1/2}(\Gamma_{N,l})^p \times L^2(\Omega_i)^p \times L^2(\Omega_i)^p \longrightarrow L_A^{-1/2}(\Gamma_{N,l})^p \end{array}$$

with :

$$\left\{ \begin{array}{rcl} S^1 & : & (\Phi_{1,r}\ ,\ f\ ,\ g) \longrightarrow A^+ W_1|_{\Gamma_{1,r}} \\ S^N & : & (\Phi_{N,l}\ ,\ f\ ,\ g) \longrightarrow A^- W_N|_{\Gamma_{N,l}} \end{array} \right.$$

for $i = 1$ and $i = N$. In the above expressions, $L_A^{-1/2}(\Gamma)$ denotes the dual space of $L_A^{1/2}(\Gamma)$. In each case W_i is the solution of the boundary value problem :

$$\left\{ \begin{array}{rcll} \mathcal{L}W_i & = & f & \text{in } \Omega_i \\ A^+ W_i & = & \Phi_{i,l} & \text{on } \Gamma_{i,l} \\ A^- W_i & = & \Phi_{i,r} & \text{on } \Gamma_{i,r} \\ A_{\boldsymbol{n}}^- W_i & = & A_{\boldsymbol{n}}^- g & \text{on } \partial\Omega \cup \partial\Omega_i \end{array} \right. \tag{19}$$

The interface operators defined previously are linear and the dependence on their arguments is done via the expressions :

$$\left\{ \begin{array}{rcl} A^- W_i|_{\Gamma_{i,l}} & = & S^i(\Phi_{i,r},0,0,0)|_{\Gamma_{i,l}} + S^i(0,\Phi_{i,l},0,0)|_{\Gamma_{i,l}} + S^i(0,0,f,g)|_{\Gamma_{i,l}} \\ A^+ W_i|_{\Gamma_i} & = & S^i(\Phi_{i,r},0,0,0)|_{\Gamma_{i,r}} + S^i(0,\Phi_{i,l},0,0)|_{\Gamma_{i,r}} + S^i(0,0,f,g)|_{\Gamma_{i,r}} \end{array} \right.$$

In order to simplify the formalism and to emphasize the contribution of each argument we define the following operators:

$$\left\{ \begin{array}{rcl} S_{rr}^i,\ S_{rl}^i & : & L_A^{1/2}(\Gamma_{i,l})^p \longrightarrow L_A^{-1/2}(\Gamma_{i,l})^p \\ S_{lr}^i,\ S_{ll}^i & : & L_A^{1/2}(\Gamma_{i,r})^p \longrightarrow L_A^{-1/2}(\Gamma_{i,r})^p \end{array} \right.$$

such that:

$$\left\{ \begin{array}{rcl} S_{rr}^i(\Phi_{i,r}) & = & S^i(\Phi_{i,r}\ ,\ 0\ ,\ 0\ ,\ 0)|_{\Gamma_{i,l}} \\ S_{rl}^i(\Phi_{i,l}) & = & S^i(0\ ,\ \Phi_{i,l}\ ,\ 0\ ,\ 0)|_{\Gamma_{i,l}} \\ S_{lr}^i(\Phi_{i,r}) & = & S^i(\Phi_{i,r}\ ,\ 0\ ,\ 0\ ,\ 0)|_{\Gamma_{i,r}} \\ S_{rl}^i(\Phi_{i,l}) & = & S^i(0\ ,\ \Phi_{i,l}\ ,\ 0\ ,\ 0)|_{\Gamma_{i,r}} \end{array} \right.$$

Within this formalism we can now reformulate symbolically the Schwarz type algorithm in terms of the fluxes $\Phi_{i,l(r)}$ defined above :

$$\Phi^{(k+1)} = \mathcal{S}(\Phi^{(k)}) + \mathcal{G} \tag{20}$$

where :

$$\Phi = (\Phi_{1,r}, ..., \Phi_{N-1,r}, \Phi_{2,l}, ..., \Phi_{N,l})^T$$

$$\mathcal{S} = \begin{bmatrix} 0 & S^2_{rr} & \dots & S^2_{rl} & \dots & 0 \\ 0 & 0 & S^3_{rr} & \dots & S^3_{rl} & 0 \\ \vdots & & & & & \vdots \\ 0 & 0 & \dots & 0 & \dots & S_N \\ S_1 & 0 & \dots & 0 & \dots & 0 \\ 0 & S^2_{lr} & \dots & S^2_{ll} & \dots & 0 \\ \vdots & & & & & \vdots \\ 0 & \dots & S^{N-1}_{lr} & \dots & S^{N-1}_{ll} & 0 \end{bmatrix} \quad \mathcal{G} = \begin{bmatrix} S^1(0,f,g)|_{\Gamma_{1,r}} \\ S^2(0,0,f,g)|_{\Gamma_{2,r}} \\ \vdots \\ S^{N-1}(0,0,f,g)|_{\Gamma_{N-1,r}} \\ S^2(0,0,f,g)|_{\Gamma_{2,l}} \\ \vdots \\ S^{N-2}(0,0,f,g)|_{\Gamma_{N-2,l}} \\ S^{N-1}(0,f,g)|_{\Gamma_{N-1,l}} \\ S^N(0,f,g)|_{\Gamma_{N,l}} \end{bmatrix}$$

The algorithm defined by (17) can be interpreted as a relaxation method applied to the system $(I - \mathcal{S})(\Phi) = \mathcal{G}$ that is a Jacobi algorithm applied to the interface system. We can also accelerate the solution of this problem by applying a Krylov type method such as GMRES [36].

In the next section, we describe the main characteristics of the mixed finite element/finite volume Euler solver designed on unstructured triangular meshes which is the starting point for the implementation of the proposed domain decomposition algorithm.

3 Numerical solution of the Euler equations

In this section we describe the characteristics of the compressible flow solver which is used as a starting point for our study. While doing so, we emphasise the aspects of particular interest to the domain decomposition approach introduced in section 2 i.e. the upwind finite volume discretization of the convective fluxes and the linearized implicit time integration strategy.

3.1 Mathematical model

Let $\Omega \subset \mathbb{R}^2$ be the computational domain and Γ its boundary. Γ is written as the union of a solid wall Γ_w and a far-field boundary Γ_∞ : $\Gamma = \Gamma_w \cup \Gamma_\infty$. Let $\vec{n}$ denotes the unitary normal at any point of Γ. The conservative form of the Euler equations is given by :

$$\frac{\partial W}{\partial t} + \vec{\nabla}.\vec{\mathcal{F}}(W) = 0 \quad , \quad W = \left(\rho, \rho\vec{U}, E\right)^T \quad , \quad \vec{\nabla} = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right)^T \tag{21}$$

where $W = W(\vec{x}, t)$; $\vec{x}$ and t respectively denote the spatial and temporal variables while $\vec{\mathcal{F}}(W) = (F_x(W)\ ,\ F_y(W))^T$ is the conservative flux whose components are given by :

$$F_x(W) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ u(E+p) \end{pmatrix} \quad , \quad F_y(W) = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ v(E+p) \end{pmatrix}$$

In the above expressions, ρ is the density, $\vec{U} = (u, v)^T$ is the velocity vector, E is the total energy per unit of volume and p is the pressure. The pressure is deduced from the other variables using the state equation for a perfect gas :

$$p = (\gamma - 1)(E - \frac{1}{2}\rho \parallel \vec{U} \parallel^2)$$

where γ is the ratio of specific heats ($\gamma = 1.4$ for the air).

3.2 Discretization is space

The flow domain Ω is discretized by a triangulation $\mathcal{T}_h$ where h is the maximal length of the edges of $\mathcal{T}_h$. A vertex of $\mathcal{T}_h$ is denoted by s_i and the set of neighboring vertices of s_i by $N(i)$. We associate to each vertex s_i a control surface (or cell) denoted by C_i which is constructed as the union of local contributions from the set of triangles sharing s_i. The contribution of a given triangle is obtained by joining its barycenter G to the midpoints I of the edges incident to s_i (see Fig. 2). The boundary of C_i is denoted by ∂C_i and the unitary normal vector exterior to ∂C_i by $\vec{\nu}_i = (\nu_{ix}, \nu_{iy})$. The union of all these cells constitutes a discretization of Ω often qualified as dual to $\mathcal{T}_h$:

$$\Omega_h = \bigcup_{i=1}^{N_V} C_i \quad , \quad N_V : \text{number of vertices of } \mathcal{T}_h$$

The spatial discretization method adopted here combines the following elements :

- a finite volume formulation together with upwind schemes for the discretization of the convective fluxes;

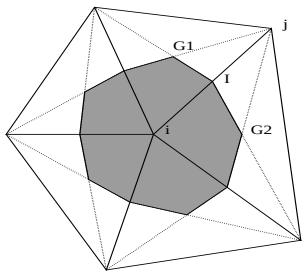


Figure 2: A control surface on a triangular mesh

- extension to second order accuracy is obtained by using the MUSCL (Monotonic Upstream Schemes for Conservation Laws) introduced by van Leer[40] and extended to unstructured triangular meshes by Fezoui and Stoufflet[12].

Integrating Eq. (21) over C_i and integrating by parts results in :

$$\begin{aligned} \iint_{C_i} \frac{\partial W}{\partial t} d\vec{x} \;+\; & \sum_{j \in N(i)} \int_{\partial C_{ij}} \vec{\mathcal{F}}(W).\vec{\nu}_i d\sigma & < 1 > \\ + \; & \int_{\partial C_i \cap \Gamma_w} \vec{\mathcal{F}}(W).\vec{n}_i d\sigma + \int_{\partial C_i \cap \Gamma_\infty} \vec{\mathcal{F}}(W).\vec{n}_i d\sigma & < 2 > \\ = \; & 0 \end{aligned} \tag{22}$$

where $\partial C_{ij} = \partial C_i \cap \partial C_j$. A first order finite volume approximation of term $< 1 >$ writes as :

$$< 1 > = \; W_i^{n+1} - W_i^n + \Delta t \sum_{j \in N(i)} \Phi_{\mathcal{F}}(W_i^n, W_j^n, \vec{\nu}_{ij}) \tag{23}$$

where $\Phi_{\mathcal{F}}$ denotes a numerical flux function such that :

$$\Phi_{\mathcal{F}}(W_i, W_j, \vec{\nu}_{ij}) \approx \int_{\partial C_{ij}} \vec{\mathcal{F}}(W).\vec{\nu}_i d\sigma \quad , \quad \vec{\nu}_{ij} = \int_{\partial C_{ij}} \vec{\nu}_i d\sigma \tag{24}$$

The numerical flux (24) yields a conservative scheme if for any edge $[s_i, s_j]$ the following condition is verified :

$$\Phi_{\mathcal{F}}(W_i, W_j, \vec{\nu}_{ij}) = -\Phi_{\mathcal{F}}(W_j, W_i, \vec{\nu}_{ji})$$

Upwinding is introduced in the calculation of (23) by using the approximate Riemann solver of Roe[35] which gives :

$$\Phi_{\mathcal{F}}(W_i, W_j, \vec{\nu}_{ij}) = \frac{\vec{\mathcal{F}}(W_i) + \vec{\mathcal{F}}(W_j)}{2}.\vec{\nu}_{ij} - \mid \mathcal{A}_R(W_i, W_j, \vec{\nu}_{ij}) \mid \frac{(W_j - W_i)}{2} \tag{25}$$

where $\mathcal{A}_R(W_i, W_j, \vec{\nu}_{ij}) = \left(\frac{\partial \vec{\mathcal{F}}}{\partial W}(W_i, W_j, \vec{\nu}_{ij}).\vec{\nu} \right)_R$ is the so-called matrix of Roe that verifies the following property :

$$\mathcal{A}_R(W_i, W_j, \vec{\nu}_{ij})(W_j - W_i) = \mathcal{F}(W_j, \vec{\nu}_{ij}) - \mathcal{F}(W_i, \vec{\nu}_{ij})$$

with :

$$\mathcal{F}(W, \vec{\nu}_{ij}) = \vec{\mathcal{F}}(W).\vec{\nu}_{ij}$$

The numerical flux (25) can thus be reformulated as :

$$\Phi_{\mathcal{F}}(W_i, W_j, \vec{\nu}_{ij}) = \mathcal{F}(W_j, \vec{\nu}_{ij}) - \mathcal{A}_R^+(W_i, W_j, \vec{\nu}_{ij})(W_j - W_i)$$

or as :

$$\Phi_{\mathcal{F}}(W_i, W_j, \vec{\nu}_{ij}) = \mathcal{F}(W_i, \vec{\nu}_{ij}) + \mathcal{A}_R^-(W_i, W_j, \vec{\nu}_{ij})(W_j - W_i)$$

with $\mathcal{A}_R(W_i, W_j, \vec{\nu}_{ij}) = A_{\vec{\nu}_{ij}}(\tilde{W})$ where $\tilde{W}$ is given by :

$$\left\{ \begin{array}{lcl} \tilde{W} & = & (\tilde{\rho}, \tilde{\rho}\tilde{u}, \tilde{\rho}\tilde{v}, \tilde{E})^T \\ \tilde{\rho} & = & (\sqrt{\rho_1}\rho_1 + \sqrt{\rho_2}\rho_2)/(\sqrt{\rho_1} + \sqrt{\rho_2}) \\ \tilde{u} & = & (\sqrt{\rho_1}u_1 + \sqrt{\rho_2}u_2)/(\sqrt{\rho_1} + \sqrt{\rho_2}) \\ \tilde{v} & = & (\sqrt{\rho_1}v_1 + \sqrt{\rho_2}v_2)/(\sqrt{\rho_1} + \sqrt{\rho_2}) \\ \tilde{H} & = & (\sqrt{\rho_1}H_1 + \sqrt{\rho_2}H_2)/(\sqrt{\rho_1} + \sqrt{\rho_2}) \end{array} \right. \tag{26}$$

where $H = \frac{\gamma p}{(\gamma - 1)\rho} + \frac{u^2 + v^2}{2}$ is the total enthalpy per unit of volume.

On the other hand, we have that :

$$\mathcal{A}^{\pm}(\tilde{W}) = \mathcal{T}(\tilde{W})\Lambda^{\pm}(\tilde{W})\mathcal{T}^{-1}(\tilde{W})$$

with :

$$\Lambda = \left(\vec{U}.\vec{\nu} - c \ , \ \vec{U}.\vec{\nu} \ , \ \vec{U}.\vec{\nu} \ , \ \vec{U}.\vec{\nu} + c \right)^T$$

where $c = \sqrt{\gamma \frac{p}{\rho}}$ denotes the speed of sound. The MUSCL technique[40] for the extension to a second order approximation of (25) relies on a linear interpolation of the state vectors W_{ij} and W_{ji} at the interface between cells C_i and C_j :

$$\tilde{W}_{ij} = \tilde{W}_i + \frac{1}{2}(\vec{\nabla}\tilde{W})_i.s_i\vec{s}_j \quad , \quad \tilde{W}_{ji} = \tilde{W}_j - \frac{1}{2}(\vec{\nabla}\tilde{W})_j.s_i\vec{s}_j \tag{27}$$

where $\tilde{W} = \left(\rho \ , \ \vec{U} \ , \ p \right)^T$ — in other words, the interpolation is done using the physical variables instead of the conservative variables. Then, the interpolated states (27) are used as arguments to the numerical flux function (23). The nodal gradients $(\vec{\nabla}\tilde{W})_i$ are obtained from a weighted average of the P1 Galerkin (centered) gradients computed on each triangle of the finite element support of s_i :

$$\left(\vec{\nabla}\tilde{W}\right)_i = \frac{\displaystyle\iint_{C_i} \vec{\nabla}\tilde{W}|_T d\vec{x}}{\displaystyle\iint_{C_i} d\vec{x}} = \frac{1}{\text{area}(C_i)} \sum_{T \epsilon C_i} \frac{\text{area}(T)}{3} \sum_{k=1, k\epsilon T}^{3} \tilde{W}_k \vec{\nabla} N_k^T \tag{28}$$

where $N_k^T(x, y, z)$ is the P1 basis function defined at the vertex s_k and associated with the triangle T. The construction given by Eq.(27)-(28) results in a half-upwind scheme which is second order accurate but can present spurious oscillations in the solution therefore expressing a loss of monotony. A classical way to cure this problem is to make a compromise between the first order and the second order schemes through the use of a slope limitation procedure[11].

3.3 Boundary conditions

The term $< 2 >$ in Eq. (22) is associated with the boundary conditions of the problem. These are now taken into account in the weak formulation. The following situations are considered :

- *solid wall.* We impose on Γ_w the slip condition $\vec{U}.\vec{n} = 0$. This condition is introduced in the corresponding term of Eq. (22) which results in :

$$\int_{\partial C_i \cap \Gamma_w} \vec{\mathcal{F}}(W).\vec{n}_i d\sigma = p_i \parallel \vec{n} \parallel \begin{pmatrix} 0 \\ \tilde{n}_{ix} \\ \tilde{n}_{iy} \\ 0 \end{pmatrix} \tag{29}$$

- *far-field boundary.* On Γ_∞, we make use of a uniform flow state vector i.e. we assume that the flow at infinity is uniform (this assumption is valid for external flows such as those considered in the results section) :

$$\rho_\infty = 1 \quad , \quad \vec{U}_\infty = (u_\infty \ , \ v_\infty)^T \quad \text{with} \quad \parallel \vec{U}_\infty \parallel = 1 \quad , \quad p_\infty = \frac{1}{\gamma M_\infty^2} \tag{30}$$

where M_∞ is the far-field Mach number. Here, an *upwind-downwind* flux decomposition is used between the external information (W_∞) and the state vector W_i associated to a vertex $s_i \in \Gamma_\infty$. More precisely, the corresponding boundary integral of term $< 2 >$ is evaluated through a non-reflexive version of the Steger and Warming flux decomposition[39] :

$$\int_{\partial C_i \cap \Gamma_\infty} \vec{\mathcal{F}}(W).\vec{n}_i d\sigma = \mathcal{A}^+(W_i, \vec{n}_{i\infty}).W_i + \mathcal{A}^-(W_i, \vec{n}_{i\infty}).W_\infty \tag{31}$$

3.4 Time integration

Assuming $W(\vec{x}, t)$ is constant on each cell C_i (in other words a mass lumping technique is applied to the temporal term in Eq. (22)) we obtain the following set of semi-discrete equations :

$$\text{area}(C_i)\frac{dW_i^n}{dt} + \Psi(W_i^n) = 0 \quad , \quad i = 1, \cdots, N_V \tag{32}$$

where $W_i^n = W(\vec{x}_i, t^n)$, $t^n = n\Delta t^n$ and :

$$\Psi(W_i^n) = \sum_{j \in N(i)} \Phi_{\mathcal{F}}(W_{ij}, W_{ji}, \vec{\nu}_{ij}) + \int_{\partial C_i \cap \Gamma} \vec{\mathcal{F}}(W).\vec{n}_i d\sigma \tag{33}$$

Explicit time integration procedures for the time integration of (32) are subject to a stability condition expressed in terms of a CFL (Courant-Friedrichs-Lewy) number. An efficient time advancing strategy can be obtained by means of an implicit linearized formulation such as the one described in Fezoui and Stoufflet[12] and briefly outlined here. First, the implicit variant of Eq. (32) writes as :

$$\frac{\text{area}(C_i)}{\Delta t^n}\delta W_i^{n+1} + \Psi(W_i^{n+1}) = 0 \quad , \quad i = 1, \cdots, N_V \tag{34}$$

where $\delta W_i^{n+1} = W_i^{n+1} - W_i^n$. Then, applying a first order linearization to the nodal flux $\Psi(W_i^{n+1})$ yields the Newton-like formulation :

$$\left(\frac{\text{area}(C_i)}{\Delta t^n} + \frac{\partial \Psi(W^n)}{\partial W}\right)\delta W^{n+1} = -\Psi(W_i^n) \tag{35}$$

In practice we replace the exact Jacobian of the second order flux $\dfrac{\partial \Psi(W^n)}{\partial W}$ by an approximate Jacobian matrix $J(W^n)$ resulting from the analytical differentiation of the first order flux (25) :

$$P(W^n)\delta W^{n+1} = \left(\frac{\text{area}(C_i)}{\Delta t^n} + J(W^n)\right)\delta W^{n+1} = -\Psi(W_i^n) \tag{36}$$

The resulting Euler implicit time integration scheme is in fact a modified Newton (see Fezoui and Stoufflet[12] for more details). As a consequence, one cannot ensure that this formulation will yield a quadratically converging method for time steps tending to infinity. The matrix $P(W^n)$ is sparse and has the suitable properties (diagonal dominance in the scalar case) allowing the use of a relaxation procedure (Jacobi or Gauss-Seidel) in order to solve the linear system of Eq. (35). Moreover, an efficient way to get second order accurate steady solutions while keeping the interesting properties of the first order upwind matrix is to use the second order elementary convective fluxes based on Eq. (27)-(28) in the right-hand side of Eq. (35). The above implicit time integration technique is well suited to steady flow calculations; for unsteady flow computations, this first order time accurate scheme is generally unacceptably dissipative.

4 Domain decomposition for the Euler equations

In this section we adapt the domain decomposition algorithm proposed in section 2 to the numerical solution of the Euler equations in the context of the spatial discretization

framework described in section 3. First, we briefly describe the basic parallelization strategy adopted in the original flow solver. Then, we introduce interface unknowns in terms of normal fluxes. The latter are first used to define a modified formulation of the global implicit system (35) permitting to distinguish purely interior unknowns from interface ones. Finally, we apply a substructuring technique to this system in order to obtain an interface system whose unknowns are defined in terms of normal fluxes.

4.1 Parallelization strategy

The parallelization strategy adopted for the single grid flow solver combines domain partitioning techniques and a message-passing programming model. This strategy has been already successfully applied in the single grid case in 2D[10] as well as in 3D[24]. The underlying mesh is assumed to be partitioned into several submeshes, each defining a subdomain. Basically the same "old" serial code is executed within every subdomain. Applying this parallelization strategy to the previously described flow solver results in modifications occuring in the main time-stepping loop and in the linear solver (which is chosen to be Jacobi in the parallel case) in order to take into account several assembly phases of the subdomain results. The assembly of the subdomain results can be implemented in one or several separated modules and optimized for a given machine. This approach enforces data locality, and therefore is suitable for all parallel hardware architectures.

For the partitioning of the unstructured mesh, two basic strategies can be considered. The first one is based on the introduction of an overlapping region at subdomain interfaces and is well suited for the mixed finite volume/element formulation considered herein. However, mesh partitions with overlapping have a main drawback : they incur redundant floating-point operations. The second possible strategy is based on non-overlapping mesh partitions and incur no more redundant floating-point operations. While updated nodal values are exchanged between the subdomains in overlapping mesh partitions, partially gathered quantities are exchanged between subdomains in non-overlapping ones. It has been our experience that both the programming effort and the performances are maximized when considering non-overlapping mesh partitions[24]. Here, according to the domain decomposition algorithm formulated in section 2, it is interesting to consider mesh partitions involving a one-triangle wide overlapping region that is shared by neighboring subdomains. As a matter of fact, it is easily seen that within this setting, the interface between two neighboring subdomains is a non-overlapping one from the viewpoint of the dual discretization of Ω in terms of control surfaces; if Ω_1 and Ω_2 are neighbors then :

$$\Gamma = \Omega_1 \cap \Omega_2 = \bigcup_{C_{1_k} \in \Omega_1, C_{2_k} \in \Omega_2} \partial C_{1_k} \cap \partial C_{2_k}$$

4.2 Domain decomposition algorithm

In order to be able to construct an interface system of the form (20), we need to consider a preliminary step which consists in the introduction of a redundant variable at the interface between two control surfaces (see Fig. 3), following a strategy adopted by Clerc[6]. Our approach is however different from the one described in [6] in the nature of this redundant variable since, as detailed below, the latter is here defined as the normal flux between two control surfaces belonging to different subdomains.

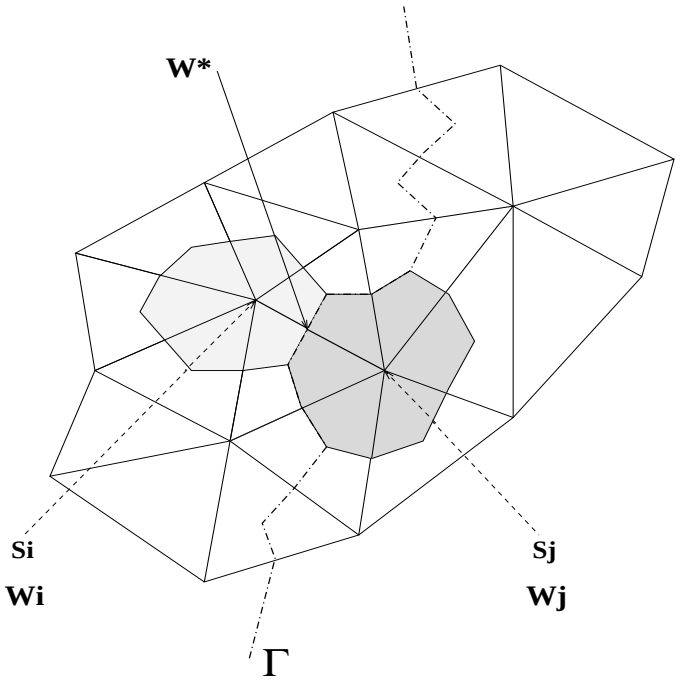


Figure 3: Definition of a redundant variable at an interface $\Gamma = \Omega_1 \cap \Omega_2$

To simplify the presentation we consider the case of a decomposition of Ω in two subdomains. Let $[s_i, s_j]$ be and edge such that C_i (associated with s_i) and C_j (associated with s_j) belong to two neighboring subdomains. An additive Schwarz formulation

is obtained by setting the following interface conditions that are taken into account in the integral formulation (22) locally in each subdomain :

$$\left\{ \begin{array}{rcl} \mathcal{A}^-_{\vec{\nu}_{ij}}(\tilde{W}^n)W_i^{(k+1)} & = & \mathcal{A}^-_{\vec{\nu}_{ij}}(\tilde{W}^n)W_j^{(k)} \\ \mathcal{A}^+_{\vec{\nu}_{ij}}(\tilde{W}^n)W_j^{(k+1)} & = & \mathcal{A}^+_{\vec{\nu}_{ij}}(\tilde{W}^n)W_i^{(k)} \end{array} \right. \tag{37}$$

where $\tilde{W}^n$ is given by (26) and we have noted $\mathcal{A}^\pm_{\vec{\nu}_{ij}}(\tilde{W}^n) = \mathcal{A}^\pm_R(W_i, W_j, \vec{\nu}_{ij})$. The above conditions are expressing the continuity of normal fluxes at the subdomain interface $\Gamma = \Omega_1 \cap \Omega_2$.

In the sequel, we simply write W_i instead of $W_i^{(k+1)}$. We introduce an auxiliary variable denoted by W^* and such that :

$$\left(\mathcal{A}^-_{\vec{\nu}_{ij}}(\tilde{W}^n)W_j\right)|_{s_j} = \left(\mathcal{A}^-_{\vec{\nu}_{ij}}(\tilde{W}^n)W^*\right)|_{\frac{s_i+s_j}{2}}$$

and

$$\left(\mathcal{A}^+_{\vec{\nu}_{ij}}(\tilde{W}^n)W_i\right)|_{s_i} = \left(\mathcal{A}^+_{\vec{\nu}_{ij}}(\tilde{W}^n)W^*\right)|_{\frac{s_i+s_j}{2}}$$

and we define :

$$\Phi = |\mathcal{A}_{\vec{\nu}_{ij}}(\tilde{W}^n)|W^* = \mathcal{A}^+_{\vec{\nu}_{ij}}(\tilde{W}^n)W_i - \mathcal{A}^-_{\vec{\nu}_{ij}}(\tilde{W}^n)W_j \tag{38}$$

the associated new unknown of the problem. We can write :

$$\Phi = \left(\mathcal{T}(\tilde{W}^n)|\Lambda(\tilde{W}^n)|\mathcal{T}^{-1}(\tilde{W}^n)\right)W^* \quad \Leftrightarrow \quad W^* = \left(\mathcal{T}(\tilde{W}^n)|\Lambda(\tilde{W}^n)|^{-1}\mathcal{T}^{-1}(\tilde{W}^n)\right)\Phi$$

where $\Lambda(\tilde{W}^n)$ is the diagonal matrix whose components are the eigenvalues of $\vec{\mathcal{F}}(W).\vec{\nu}_{ij}$ and $\mathcal{T}(\tilde{W}^n)$ is the matrix whose columns are the associated left eigenvectors. The positive and negative parts of this flux are given by :

$$\begin{array}{rcl} \Phi^\pm & = & \mathcal{A}^\pm_{\vec{\nu}_{ij}}(\tilde{W}^n)W^* \\ & = & \left(\mathcal{T}(\tilde{W}^n)\Lambda^\pm(\tilde{W}^n)\mathcal{T}^{-1}(\tilde{W}^n)\right)W^* \\ & = & \left(\mathcal{T}(\tilde{W}^n)\Lambda^\pm(\tilde{W}^n)|\Lambda^{-1}(\tilde{W}^n)|\mathcal{T}^{-1}(\tilde{W}^n)\right)\Phi \end{array} \tag{39}$$

that we write in condensed form as :

$$\Phi^{\pm} = P^{\pm}(\tilde{W}^n)\Phi$$

On the other hand, the elementary fluxes associated with the control surfaces C_i and C_j can be expressed in terms of the auxiliary flux Φ as :

$$\left\{ \begin{array}{rcl} \Phi^i(W_i, W^\star, \vec{\nu}_{ij}) & = & \left(\mathcal{A}_{\vec{\nu}_{ij}}(W_i^n) - \mathcal{A}^-_{\vec{\nu}_{ij}}(\tilde{W}^n)\right) W_i + \mathcal{A}^-_{\vec{\nu}_{ij}}(\tilde{W}^n) W^* \\ & = & \left(\mathcal{A}_{\vec{\nu}_{ij}}(W_i^n) - \mathcal{A}^-_{\vec{\nu}_{ij}}(\tilde{W}^n)\right) W_i + P^-(\tilde{W}^n)\Phi \\ \Phi^j(W^\star, W_j, \vec{\nu}_{ij}) & = - & \left(\mathcal{A}_{\vec{\nu}_{ij}}(W_j^n) - \mathcal{A}^+_{\vec{\nu}_{ij}}(\tilde{W}^n)\right) W_j - \mathcal{A}^+_{\vec{\nu}_{ij}}(\tilde{W}^n) W^* \\ & = - & \left(\mathcal{A}_{\vec{\nu}_{ij}}(W_j^n) - \mathcal{A}^+_{\vec{\nu}_{ij}}(\tilde{W}^n)\right) W_j - P^+(\tilde{W}^n)\Phi \end{array} \right. \tag{40}$$

Taking into account Eq. (38) and (40) we can construct an implicit linear system that distinghishes purely interior unknowns (state vectors) from interface ones (normal fluxes) :

$$\left(\begin{array}{ccc} \mathcal{M}_1 & 0 & \mathcal{M}_{12} \\ 0 & \mathcal{M}_2 & \mathcal{M}_{21} \\ \mathcal{F}_1 & \mathcal{F}_2 & \mathrm{I}d \end{array} \right) \left(\begin{array}{c} W_1 \\ W_2 \\ \Phi \end{array} \right) = \left(\begin{array}{c} b_1 \\ b_2 \\ 0 \end{array} \right) \tag{41}$$

where $\mathcal{M}_1$ (respectively $\mathcal{M}_2$) is the matrix that couples the unknowns associated with vertices internal to Ω_1 (respectively Ω_2) whereas $\mathcal{F}_1$, $\mathcal{F}_2$, $\mathcal{M}_{12}$ and $\mathcal{M}_{21}$ are coupling matrices between internal and interface unknowns. These various matrix terms are detailed in annex 7. Now, the internal unknowns can be eliminated in favor of the interface ones to yield the following interface system :

$$\begin{array}{rcl} S\Phi & = & [Id - (\mathcal{F}_1\mathcal{M}_1^{-1}\mathcal{M}_{12} + \mathcal{F}_2\mathcal{M}_2^{-1}\mathcal{M}_{21})]\Phi \\ & = & g \\ & = & -[\mathcal{F}_1\mathcal{M}_1^{-1}b_1 + \mathcal{F}_2\mathcal{M}_2^{-1}b_2] \end{array} \tag{42}$$

As usual in this context, once this system has been solved for Φ, we obtain the values of the purely internal unknowns by performing independent (i.e. parallel) local solves :

$$\left\{ \begin{array}{rcl} W_1 & = & \mathcal{M}_1^{-1}(b_1 - \mathcal{M}_{12}\Phi) \\ W_2 & = & \mathcal{M}_2^{-1}(b_2 - \mathcal{M}_{21}\Phi) \end{array} \right. \tag{43}$$

A simple algorithm for solving system (42) is given by the following Richardson type iteration.

ALGORITHM 1 (**RICH**) *: Richardson type iteration for solving the interface system* $S\Phi = g$.

- *Initialisation :* $\Phi = \Phi^\circ$
- *Computation of* $g = g_1 + g_2$ *(including communication steps to assemble local contributions) with :*
  $g_i = \mathcal{F}_i x_i$ *where* x_i *is obtained through the local solution* : $\mathcal{M}_i x_i = b_i$
- *Main parallel loop :* $k = 0, .., K$
  - *Subdomain* Ω_1 : $y_1 = \mathcal{M}_{12}\Phi^k$ *and* $\Phi_1 = \mathcal{F}_1 v_1$
    where v_1 *is obtained through the local solution :* $\mathcal{M}_1 v_1 = y_1$
  - *Subdomain* Ω_2 : $y_2 = \mathcal{M}_{21}\Phi^k$ *and* $\Phi_2 = \mathcal{F}_2 v_2$
    where v_2 *is obtained through the local solution :* $\mathcal{M}_2 v_2 = y_2$
  - *Assembly process (including communication steps) :* $\Phi^{k+1} = \Phi_1 + \Phi_2 + g$
- *If* $\|\Phi^{k+1} - \Phi^k\| < \epsilon$ *then exit main loop*

5 Solution strategy for the local problems

The domain decomposition algorithm proposed in section 4 calls for independent (parallel) local solution steps in each subdomain. Here, we are interested in solving the corresponding linear systems iteratively, the main reasons being that, on one hand, direct solvers are characterized by high memory and CPU requirements and, on the other hand, we would like to study (at least experimentally) the influence of approximate local solutions on the convergence of the overall domain decomposed flow solver. Two iterative solution strategies have been considered :

- a *single grid* strategy relying on the Jacobi relaxation method. In the original solver, this method is used to approximately solve the linear system (35). Here, the Jacobi method will be employed as a local iterative solver for the sub-problems resulting from the application of the substructuring technique to system (41);
- a *multigrid* strategy applied at the subdomain level. In this case the Jacobi method (or the Gauss-Seidel method) is used as a smoother and the multigrid method is the accelerator. The remaining of this section is devoted to a description of the adopted multigrid strategy.

In addition, the Jacobi method as a global (parallel) iterative solver will also be used to solve the linear system (41).

5.1 Linear multigrid strategy

The multigrid method considered in the present study aims at accelerating the iterative solution of linear systems resulting from the adoption of a linearized implicit scheme for time advancing the equations modelling compressible fluid flows. It is well known that classical relaxation methods such as the Jacobi or Gauss-Seidel methods, applied to the iterative solution of the resulting linear system, quickly damp the high frequencies of the error however they do not allow for an efficient treatment of the low frequency components. The basic idea of the coarse grid correction scheme is to transfer the partially solved solution on a coarser grid in order to transform the low frequencies of the fine grid solution in high frequencies which are then efficiently damped by the standard relaxation methods.

We describe below the various steps of the linear multigrid algorithm.

ALGORITHM 2 *(***MGV***) : linear multigrid, V-cycle for the solution of* $\mathcal{M}x = b$.

The following notations are adopted :

- G_m *: m-th grid of the multigrid hierarchy (*$m = 1$ *is the finest level and* $m = M$ *is the coarsest level).*
- x^α *: solution vector after the* α*-th multigrid V-cycle (*x^α *is an approximation of the exact fine grid solution* x*) with* x^0*=0.*
- $x^0_{(m)}$ *: initial iterate for the relaxation method used as a smoother for the linear system built on level* G_m*, for* $m = 1, ..., M$.
- $p_{m+1,m}$ *and* $r_{m,m+1}$ *are the prolongation and restriction operators (these inter-grid transfer operators are defined later) between grids* G_m *and* G_{m+1}.
- ν_1 *and* ν_2 *are the number of relaxation steps, later referred as the number of pre- and post-smoothing steps.*
- ω *is the relaxation parameter of the iterative method.*
- $\mathcal{M}_{(m)}$ *is the discrete operator on grid* G_m.
- $b_{(m)}$ *is the right-hand side of the linear system built on grid* G_m.

- *Initialisation :* $x^0_{(1)} = x^\alpha$
- *Ascending phase : calculation of the coarse grid corrections*
  - *Loop from level* $m = 1$ *(finest level) to the (coarse) level* $m = M - 1$ *:*

$$x^0_{(m)} = \begin{cases} \mathcal{M}^{-1}_{ii_{(m)}} b_{i(m)}, & \text{if } m > 1 \\ x^\alpha, & \text{if } m{=}1 \end{cases}$$

- for the Jacobi relaxation method as a smoother

$$\overline{x^l_{i(m)}} = \mathcal{M}^{-1}_{ii_{(m)}} \left[b_{i(m)} - \sum_{j \neq i} \mathcal{M}_{ij_{(m)}} x^{l-1}_{j(m)} \right] \quad , \quad l = 1, ..., \nu_1$$

- for the Gauss-Seidel method as a smoother

$$\overline{x^l_{i(m)}} = \mathcal{M}^{-1}_{ii_{(m)}} \left[b_{i(m)} - \sum_{j<i} \mathcal{M}_{ij_{(m)}} x^l_{j(m)} - \sum_{j>i} \mathcal{M}_{ij_{(m)}} x^{l-1}_{j(m)} \right] \quad , \quad l = 1, ..., \nu_1$$

$$\begin{cases} x^l_{i(m)} & = & (1-\omega)\, x^{l-1}_{i(m)} + \omega \overline{x^l_{i(m)}} \\ b_{i(m+1)} & = & r_{m,m+1} \left(b_{i(m)} - \sum\limits_{j \neq i; j=i} \mathcal{M}_{ij_{(m)}} x^{\nu_1}_{j(m)} \right) \end{cases}$$

  - *End of the ascending phase*
- *Descending phase : prolongation of the coarse grid corrections*
  - *Loop from level* $m = M$ *(coarsest level) to level* $m = 1$ *(finest level) :*

$$\begin{cases} x_{i(m)} & = & x^{\nu_1}_{i(m)} + p_{m+1,m} \left(x_{i(m+1)} \right) \\ x^0_{i(m)} & = & x_{i(m)} \end{cases}$$

- for the Jacobi relaxation method as a smoother

$$\overline{x^{l}_{i(m)}} = \mathcal{M}^{-1}_{ii(m)} \left[b_{i(m)} - \sum_{j \neq i} \mathcal{M}_{ij(m)} x^{l-1}_{j(m)} \right] \quad , \quad l = 1, ..., \nu_2$$

- for the Gauss-Seidel method as a smoother

$$\overline{x^{l}_{i(m)}} = \mathcal{M}^{-1}_{ii(m)} \left[b_{i(m)} - \sum_{j<i} \mathcal{M}_{ij(m)} x^{l}_{j(m)} - \sum_{j>i} \mathcal{M}_{ij(m)} x^{l-1}_{j(m)} \right] \quad , \quad l = 1, ..., \nu_2$$

$$\left\{ \begin{array}{rcl} x^{l}_{i(m)} & = & (1-\omega)\, x^{l-1}_{i(m)} + \omega \overline{x^{l}_{i(m)}} \\ x_{i(m)} & = & x^{\nu_2}_{i(m)} \end{array} \right.$$

– *End of the descending phase*

- *Update of the multigrid solution :* $x_i{}^{\alpha+1} = x_{i(1)}$

5.2 Grid coarsening by agglomeration

The multigrid strategy adopted here is based on the use of macro elements (macro control surfaces) which form the coarse discretizations of the computational domain. Such a strategy has been considered by several authors for the solution of the Euler equations (see Lallemand *et al.*[23]), the Navier-Stokes equations (see Mavriplis and Venkatakrishnan[26]) and the Reynolds Average Navier-Stokes equations (see Mavriplis and Venkatakrishnan[27], Carré[5] and Mavriplis[25]. In [23] the adopted coarsening algorithm is based on neighboring relations. Starting from a fine unstructured triangulation, one wants to generate a hierarchy of coarse levels ; this can be achieved using a "greedy" type coarsening algorithm that assembles neighboring control volumes of the finest grid (e.g. those having a common boundary) to build the macro elements of the coarser level. The main advantage of this method is that it allows for an automatic generation of the coarser discretizations without building any coarse triangulation. This algorithm is illustrated on Fig. 4.

5.3 Coarse grid approximation for convective terms

Recall that the convective fluxes are integrated between two control volumes of the finest mesh ; they are computed in the same way on a coarse level, between two macro

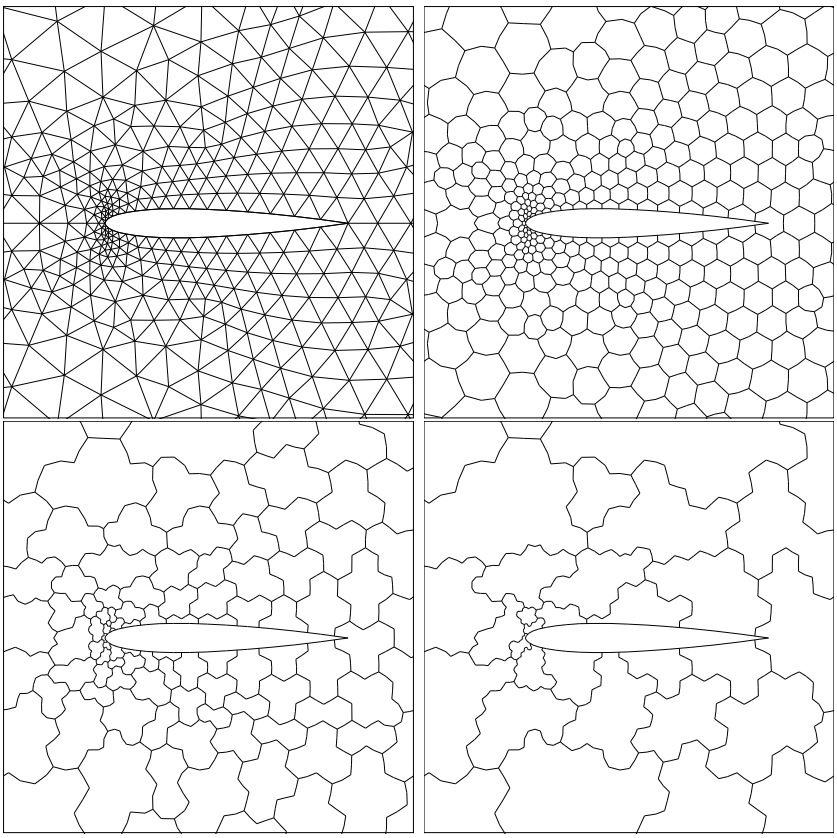


Figure 4: Illustration of the grid coarsening by agglomeration
Top left : triangular grid - Top right : dual grid
Bottom left : first agglomerated level
Bottom right : second agglomerated level

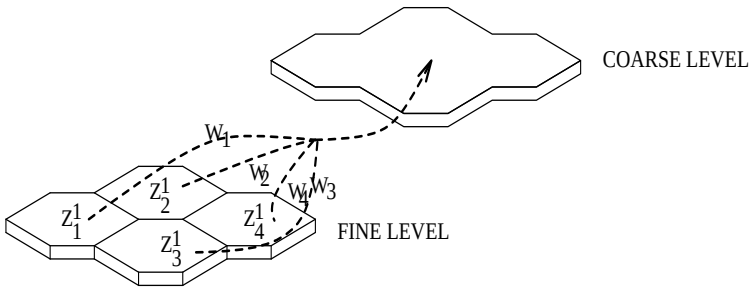
elements. However, on the coarse grids, this computation is limited to first order accuracy because nodal gradients cannot be evaluated as they are on the finest mesh; this is really not a problem here as the multigrid method is only used for the acceleration of a linear system solution based on first order Jacobian (implicit) matrices. Both conservative variables and normal vectors are interpolated between the different grids. The coarse grid variables are deduced by transfer operators (defined later). The normal vectors, linked with each coarse mesh macro element, result from the summation of the finer grid vectors (for the fine mesh control surfaces that have a common boundary with the coarse mesh macro element) ; as a result, at most one flux is computed between two macro control volumes.

5.4 Inter-grid transfer operators

A condition to obtain multigrid efficiency is that the summation of the orders of the transfer operators is greater than the order of the partial differential equation to be solved[17]. For instance, this condition, developed in [41] and [21], requires in order to solve the Navier-Stokes equations, that either prolongation or restriction be linear. However, a linear interpolation is not easily built in the agglomeration context. For the solutions of the Euler equations, we keep the same order for both restriction and prolongation which is compatible with the previous condition for the purely convective approximation, and allows building simple and diagonally dominant coarse grid matrices.

We define $[W_m]$ as the solution vector on the m-th level where $[W_m]\,(Z_l^m)$ denotes the vector of conservative variables in the l-th cell (Z_l^m) of grid level G_m ($l = 1, \cdots, N_m$, where N_m is the number of cells on the m-th level).

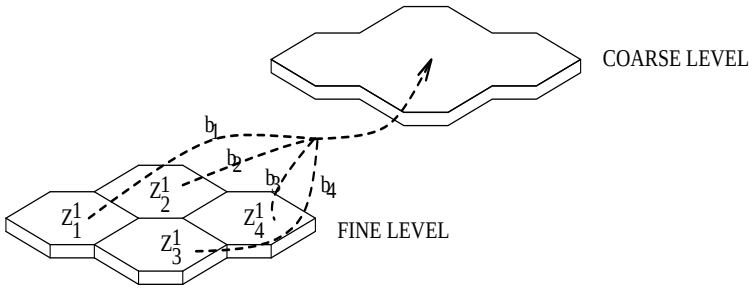
- *Restriction operator* : $r^s_{m,m+1}$. This operator is used for transferring the solution $[W_m]$.



$$\text{For } j = 1, \cdots, N_{m+1} \quad : \quad \left[r^s_{m,m+1}\left(W_m\right)\right]\left(Z^{m+1}_j\right) = \frac{\sum\limits_{l \in U^m(j)} \text{area}\left(Z^m_l\right) W_m\left(Z^m_l\right)}{\text{area}\left(Z^{m+1}_j\right)}$$

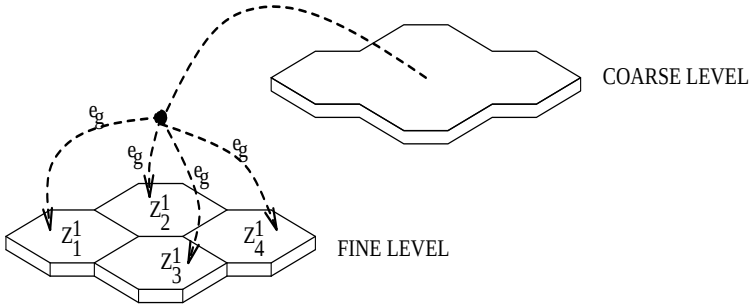
where $U^m(j)$ denotes the set of the indices l of Z^m_l zones on the level m with $Z^{m+1}_l = \bigcup\limits_{l \in U^m(j)} Z^m_l$. This operator transfers the solution from level m to level $m+1$ in order to build only the discretization of the continuous operator on this level and is not used in the multigrid cycle. It is a weighted approximation of the level m solution.

- *Restriction operator* : $r^{rhs}_{m,m+1}$. This operator is used for transferring the right-hand side $[b_m]$. The RHS on level $m+1$ is obtained by summation of RHS from level m.



$$\text{For } j = 1, \cdots, N_{m+1} \quad : \quad \left[r^{rhs}_{m,m+1}\left(b_m\right)\right]\left(Z^{m+1}_j\right) = \sum_{l \in U^m(j)} b_m\left(Z^m_l\right)$$

- *Prolongation operator* : $p^s_{m+1,m}$. This operator is used for transferring the correction $[e_{m+1}]$. from level $m+1$ to level m. It is defined by a trivial injection.



$$\text{For } l = 1, \cdots, N_m \quad : \quad \left[p^s_{m+1,m}\left(e_{m+1}\right)\right]\left(Z^m_j\right) = e_{m+1}\left(Z^{m+1}_l\right)$$

- **Restriction operator** for the solution vectors W_m : $r^s_{m,m+1}$ $(j = 1, \cdots, N_{m+1})$

$$\left[r^s_{m,m+1}\left(W_m\right)\right]\left(Z_j^{m+1}\right) = \frac{\sum\limits_{l \in U^m(j)} \mathrm{vol}\left(Z_l^m\right) W_m\left(Z_l^m\right)}{\mathrm{vol}\left(Z_j^{m+1}\right)} \tag{44}$$

where $U^m(j)$ denotes the set of the l indices of the macro elements Z_l^m on the level m with $Z_l^{m+1} = \bigcup\limits_{l \in U^m(j)} Z_l^m$. This operator is used to transfer the solution from level m to level $m+1$ in order to build only the discretization of the continuous operator on this level and is not used in the multigrid cycle. It is a weighed approximation of the solution on level m.

- **Restriction operator** for the right hand side : $r^{rhs}_{m,m+1}$. The RHS on level $m+1$ is obtained by summation of RHS from level m. It is defined as :

$$\left[r^{rhs}_{m,m+1}\left(b_m\right)\right]\left(Z_j^{m+1}\right) = \sum_{l \in U^m(j)} b_m\left(Z_l^m\right) \tag{45}$$

- **Prolongation operator** for the error/correction term : $p^c_{m+1,m}$. This operator transfers the correction e_{m+1} from level $m+1$ to level m. It is defined by a trivial injection between the two grids :

$$\left[p^c_{m+1,m}\left(e_{m+1}\right)\right]\left(Z_j^m\right) = e_{m+1}\left(Z_l^{m+1}\right) \quad , \quad l = 1, \cdots, N_m \tag{46}$$

6 Numerical results

6.1 Test case definition

The test case under consideration is given by the flow around the `NACA0012` airfoil. Three unstructured triangular meshes have been used whose characteristics are given in Tab. 1 (see Fig. 5 for a partial view of mesh N1). Mesh N2 and N3 have been obtained by uniform division of mesh N1.

The following situations have been considered :

Table 1: Characteristics of the meshes for the `NACA0012` airfoil

Mesh	# Vertices	# Triangles	# Edges
N1	3114	6056	9170
N2	12284	24224	36508
N3	48792	96896	145688

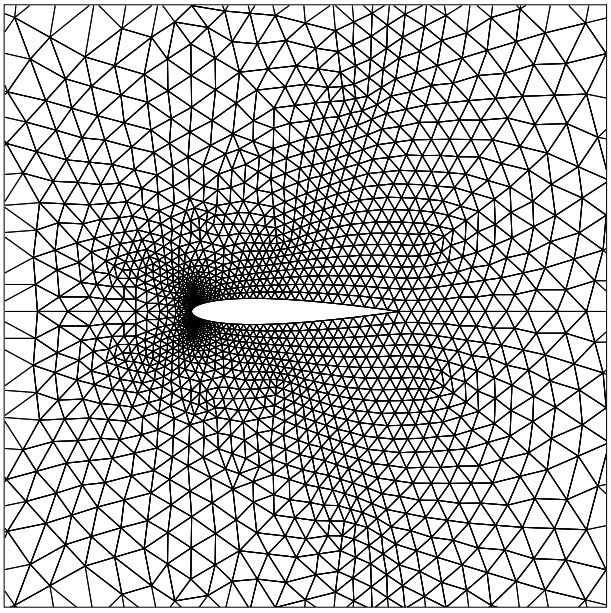


Figure 5: Unstructured triangular mesh around the `NACA0012` airfoil

S1 : the subsonic flow at a free stream Mach number equal to 0.3 and an angle of attack of 0°. In that case, the extension to second order accuracy in space does not make use of a limiter. The time step is obtained using constant values of the CFL number. The value CFL=1000 has been used for the steady flow computation however, as will be seen in the sequel, we have also investigated the influence of the value of the CFL number on the convergence of the domain decomposition solver.

S2 : the transonic flow at a free stream Mach number equal to 0.85 and an angle of attack of 0°. In that case, the time step is obtained using the law CFL=$5 \times k_t$ where k_t denotes the time iteration. The Van Albada limiter is used in the MUSCL technique (see Fezoui and Dervieux[11] for more details).

6.2 Computing platforms and conventions

Numerical experiments have been performed on a cluster of 12 `Pentium Pro 200 Mhz` computers (running the LINUX system) interconnected via a 100 Mbit/s `FastEthernet` switch. The MPI implementation is MPICH. The code is written in FORTRAN 77 and the GNU G77 compiler has been used with maximal optimization options.

Performance results are given for 64 bit arithmetic computations. In the following tables, N_p is the number of processes for the parallel execution, N_g is the total number of levels in the multigrid hierarchy (fine mesh included), N_c denotes the number of multigrid cycles used for each linear system solution; "Elapsed" denotes the total elapsed execution time and "CPU" denotes the total CPU time (taken as the maximum value over the local measures); "% CPU" denotes the ratio of "CPU" to "Elapsed" i.e. this ratio gives an idea of the CPU utilization. This ratio is our principal measure of parallel efficiency. The difference between "Elapsed" and "CPU" basically yields the sum of the communication and idle times, the latter being related to computational load unbalance. The parallel speedup $S(N_p)$ is always calculated using the elapsed execution times.

6.3 Interface system solvers

In order to solve the linear system $S\Phi = g$ (see Eq. (42)) we have considered the following three strategies :

- a simple Richardson type iteration (see algorithm 1);
- a full GMRES iteration[36];
- a left preconditioned full GMRES iteration based on a simple algebraic polynomial preconditioner briefly described below. It is clear that the objective here is not to devise and apply an optimal preconditioner for the interface system (this is actually the object of an ongoing effort) but rather to have a preliminary evaluation of the role of a preconditioner in the present context.

The expression of the interface matrix of Eq. (42) can be written as :

$$S = Id - (\mathcal{F}_1 \mathcal{M}_1^{-1} \mathcal{M}_{12} + \mathcal{F}_2 \mathcal{M}_2^{-1} \mathcal{M}_{21}) = Id - A$$

then the construction of the preconditioner starts from the fact that the matrix A results from the discretisation of a contractant operator[13] therefore we have that :

$$(Id - A)^{-1} = Id + A + A^2 + ... \tag{47}$$

An approximate inverse and thus a preconditioner for S is given by :

$$S' = Id + A \tag{48}$$

The preconditioned form of (42) writes as :

$$S'S\Phi = S'g \Leftrightarrow (Id - A^2)\Phi = g' \tag{49}$$

Clearly the above preconditioner is not computationally cheap since it requires one additional matrix vector product using matrix A within each GMRES iteration, therefore its application will hardly results in a reduction in the execution time; hence, we will mainly assess here the effect of this preconditioner on the required number of GMRES iterations.

6.4 Solution of the first linear system

The following series of numerical experiments concentrates on the solution of the linear system resulting from the first implicit time step starting from a uniform flow. If not explicitly stated otherwise, the linear thresholds for the local (ε_l) and the interface (ε_i) system solutions are given by $\varepsilon_l = \varepsilon_i = 10^{-10}$.

6.4.1 Subsonic flow test case

We begin with numerical experiments for the subsonic flow test case using the value CFL=20. Fig. 6 to 8 compare the convergence of the interface system solution for each mesh of Tab. 1 and for $N_p = 8$. The effective number of iterations to convergence are summarized in Tab. 2, including results for various decompositions. Timings for mesh N3 and $N_p = 12$ are given in Tab. 3. Several comments can be made :

- the convergence of a domain decomposition algorithm is generally assessed with regards to two factors : the number of degrees of freedom i.e. the characteristic dimension h of the underlying mesh and the number of subdomains i.e. the characteristic dimension $H \approx 1/N_p$ of the domain decomposition (each subdomain being viewed as a macro-element). The results of Tab. 2 show that the convergence of the proposed domain decomposition algorithm is rather insensitive to h whereas the dependence on H is more important;

- following the previous comment, one might legitimately question about the influence of the problem stiffness on the convergence of the proposed domain decomposition solver. In the present context, this can be assessed by increasing the CFL number (the limit should be given by the maximum value that allows a correct calculation of the steady flow i.e. without incurring negative values of the density or the pressure). Fig. 9 compares the convergence of the interface system solution for several values of the CFL number using mesh N3 and for $N_p = 8$ while Fig. 10 visualises the convergence for each mesh of Tab. 1 using the value CFL=1000 and for $N_p = 8$; in this case, the number of iterations increases from 56 (mesh N1) to 73 (mesh N3). On the other hand, Fig. 11 visualises the influence of the decompositions of mesh N3 for CFL = 1000. The dependence of the convergence on the parameter H is made clearer on this last figure : whereas the number of iterations increased from 24 ($N_p = 2$) to 28 ($N_p = 24$) in the case CFL=20 (see Tab. 2), it is increasing from 57 ($N_p = 4$) to 98 ($N_p = 24$) for CFL=1000. The influence of the simple algebraic preconditioning adopted here is depicted on Fig. 12 : the number of iterations is now increasing from 30 ($N_p = 4$) to 52 ($N_p = 24$). The main effect of this preconditioner is to reduce the number of iterations by a factor close to 2 (1.88 for $N_p = 24$) however is does not improve the scalability of the domain decomposed solver. As expected, since the application of the preconditioner basically requires one additional matrix vector with the original matrix, the gain in the number of iterations does not translate into a gain in execution times;

- it is clear that highly accurate solutions of the local systems result in computational costs that are prohibitively high. Tab. 3 shows that in the best case, the domain decomposition approach is about 13 times more expensive than the global solution strategy based on the simple Jacobi solver. From this table it is also seen that the domain decomposition solver demonstrates higher parallel efficiencies as illustrated by the values of the "%CPU" measure. Clearly, this behavior is the result of reduced communication costs as compared to what is induced by the global Jacobi solver;

- there are at least two strategies that can be considered for reducing the overall cost of the domain decomposition solver. The first one consists in adopting a more efficient local solver. Here this has been achieved by using the multigrid acceleration described in section 5 : a pointwise Gauss-Seidel method has been selected as the smoother in the context of a V-cycle with $\nu_1 = \nu_2 = 2$ and using 3 coarse grid levels (i.e. $N_g = 4$). Tab. 4 compares the single grid (SG) and the multigrid (MG) local iterative solvers for mesh N3 and $N_p = 8$. Second, one may ask if it is really necessary to solve accurately the subdomain problems.

A partial answer is given here on Fig. 13 where we compare the convergence of the full GMRES solver for mesh N3, $N_p = 8$ and for two values of the local linear threshold $\varepsilon_l = 10^{-2}$ and $\varepsilon_l = 10^{-10}$. From these figures and the timings in Tab. 4 we note that inexact local solutions are not affecting the convergence of the GMRES method and result in a notable reduction in the overall cost of the domain decomposition solver. Of course, the validity of this strategy should be assessed in details in the context of more challenging situations such as the calculation of unsteady flows.

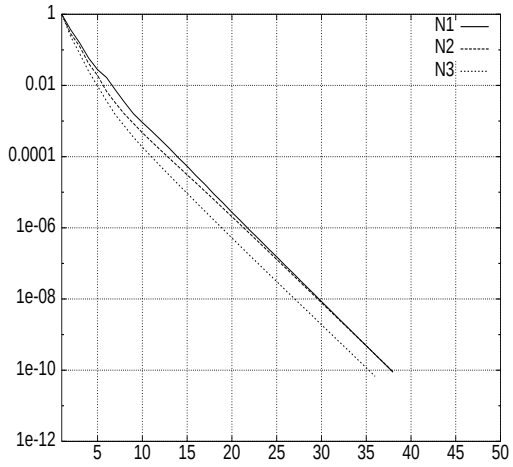


Figure 6: Subsonic flow test case: convergence of the first linear system (CFL = 20) Solution of the interface system : Richardson type iteration ($N_p = 8$)

6.4.2 Transonic flow test case

Here, we report on results of numerical experiments for the transonic flow test case using the value CFL=5 (for this particular test case, the calculation of the steady flow requires to start with smaller values of the CFL number compared to what was possible with the subsonic test case). Fig. 14 to 16 compare the convergence of the interface system solution for each mesh of Tab. 1 and for $N_p = 8$. The effective number of iterations to convergence are summarized in Tab. 5, including results for various decompositions. These results are similar to those obtained for the subsonic test case and do not call for particular remarks.

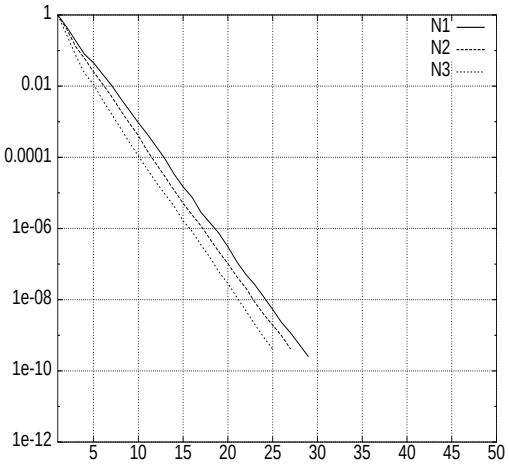


Figure 7: Subsonic flow test case: convergence of the first linear system (CFL = 20)
Solution of the interface system : full GMRES iteration ($N_p = 8$)

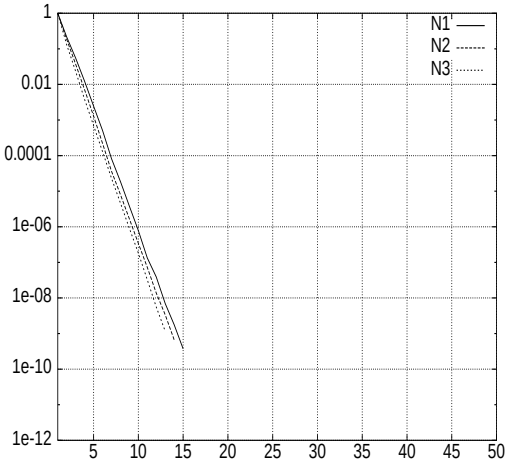


Figure 8: Subsonic flow test case: convergence of the first linear system (CFL = 20)
Solution of the interface system : left preconditioned full GMRES iteration ($N_p = 8$)

Table 2: Convergence of the first linear system (CFL = 20)
Subsonic flow test case

	N_p	N1	N2	N3
Jacobi (global system)	2	322	358	453
Richardson	2	31	33	32
	4	31	32	34
	8	38	38	36
	12	-	-	38
	24	-	-	38
GMRES	2	21	23	24
	4	23	24	24
	8	29	27	25
	12	-	-	28
	24	-	-	28
Preconditioned GMRES	2	11	12	13
	4	12	12	13
	8	15	14	13
	12	-	-	14
	24	-	-	14

Table 3: Timings for the solution of the first linear system (CFL =20)
Subsonic flow test case : mesh N3

Method	N_p	CPU	Elapsed	% CPU
Jacobi (global system)	12	61 sec	65 sec	94.0
Richardson	12	1058 sec	1074 sec	98.5
GMRES	12	832 sec	846 sec	98.3
Preconditioned GMRES	12	846 sec	862 sec	98.1

Table 4: Timings for the solution of the first linear system (CFL = 20)
Subsonic flow test case : mesh N3
Comparison of local solution strategies

Interface system	Local system	N_p	CPU	Elapsed	% CPU
GMRES	SG/$\varepsilon_l = 10^{-10}$	8	1575 sec	1601 sec	98.3
	MG/$\varepsilon_l = 10^{-10}$	8	496 sec	511 sec	97.1
	MG/$\varepsilon_l = 10^{-2}$	8	137 sec	142 sec	96.5

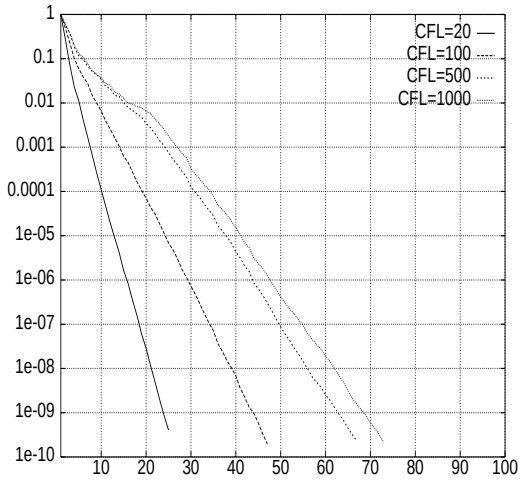


Figure 9: Subsonic flow test case: convergence of the first linear system (mesh N3)
Solution of the interface system : full GMRES iteration ($N_p = 8$)
Influence of the CFL number

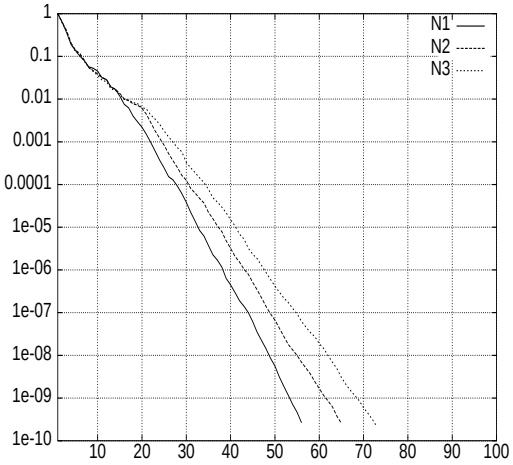


Figure 10: Subsonic flow test case: convergence of the first linear system (CFL = 1000)
Solution of the interface system : full GMRES iteration ($N_p = 8$)

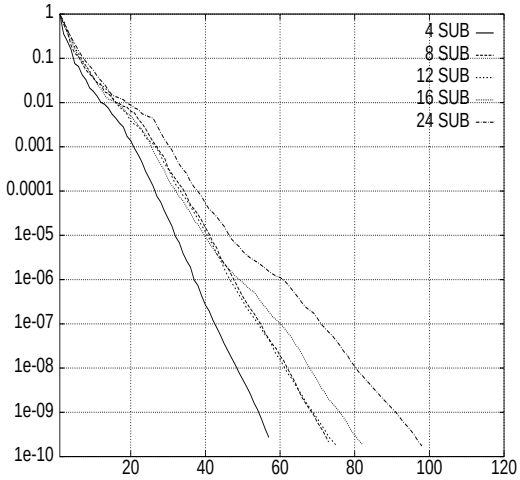


Figure 11: Subsonic flow test case: convergence of the first linear system
Solution of the interface system : full GMRES iteration (CFL = 1000)
Influence of the number of subdomain (mesh N3)

Table 5: Convergence of the first linear system (CFL = 5)
Transonic flow test case

	N_p	N1	N2	N3
Jacobi (global system)	2	322	358	453
Richardson	2	21	22	22
	4	21	22	22
	8	23	23	22
	16	-	-	22
	24	-	-	22
GMRES	2	17	18	18
	4	16	17	18
	8	17	18	18
	16	-	-	18
	24	-	-	19
Preconditioned GMRES	2	9	9	9
	4	9	9	9
	8	9	9	9
	16	-	-	9
	24	-	-	10

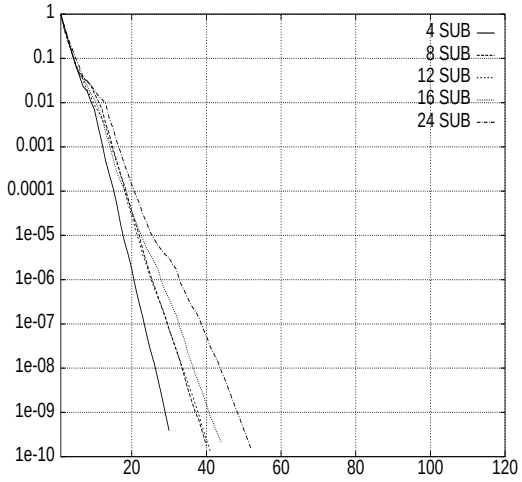


Figure 12: Subsonic flow test case: convergence of the first linear system (CFL = 1000)
Solution of the interface system : left preconditioned full GMRES iteration
Influence of the number of subdomain (mesh N3)

6.5 Full steady flow calculations

6.5.1 Subsonic flow test case

Steady iso-Mach lines for this test case are visualized on Fig. 17. The results presented below are all characterized by the following points :

- calculations are performed using mesh N3;
- the CFL number is set to a constant value of 1000;
- the calculation starts from a uniform flow;
- the local systems induced by the domain decomposition solver are never solved with a high accuracy.

One objective of this section was to verify that the last option above did not influence the convergence to the steady state compared to the reference convergence as produced by the global implicit approach (see section 3.4) where the linear system (36) is approximately solved using Jacobi relaxations. For instance, Fig. 18 visualises

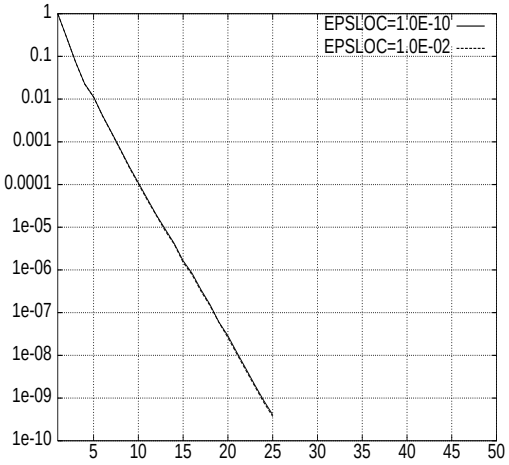


Figure 13: Subsonic flow test case: convergence of the first linear system
Solution of the interface system : full GMRES iteration ($N_p = 8$)
Influence of the accuracy of subdomain solutions (CFL = 20, mesh N3)

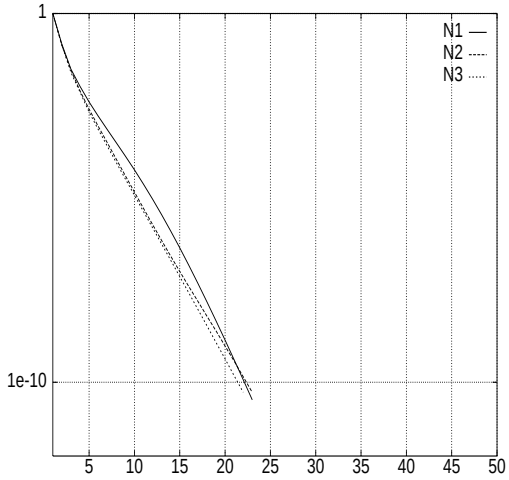


Figure 14: Transonic flow test case: convergence of the first linear system (CFL = 5)
Solution of the interface system : Richardson type iteration ($N_p = 8$)

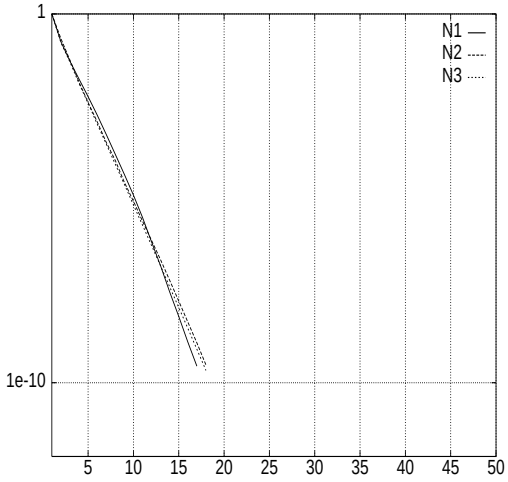


Figure 15: Transonic flow test case: convergence of the first linear system (CFL = 5)
Solution of the interface system : full GMRES iteration ($N_p = 8$)

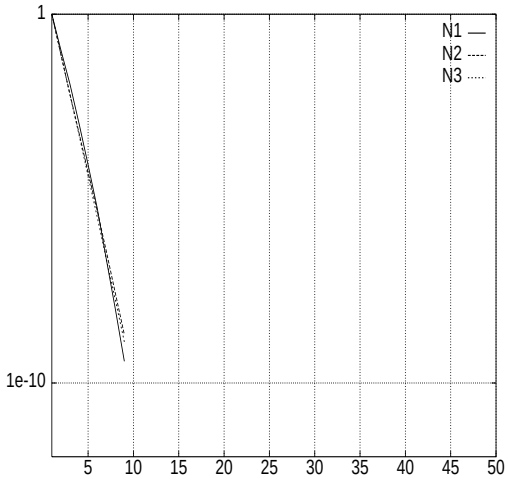


Figure 16: Transonic flow test case: convergence of the first linear system (CFL = 5)
Solution of the interface system : left preconditioned full GMRES iteration ($N_p = 8$)

the non-linear convergence in terms of the normalised energy residual for the following situations :

- the global solution strategy where, at each time step, the linear system (36) is solved using Jacobi relaxations with a linear threshold fixed to $\varepsilon_g = 10^{-1}$;
- the proposed DDM strategy where, at each time step, the interface system (42) is solved using full GMRES iterations (without preconditioning) with a linear threshold fixed to $\varepsilon_i = 10^{-1}$; moreover, the local linear systems are solved using either 1 V-cycle using $\nu_1 = 4$ pre-smoothing and $\nu_2 = 4$ post-smoothing steps, or 4 V-cycles using $\nu_1 = 2$ pre-smoothing and $\nu_2 = 2$ post-smoothing steps (in both cases the smoother is a pointwise Gauss-Seidel method) using 3 coarse grid levels (i.e. $N_g = 4$).

Effective number of time steps to convergence (initial normalised energy residual reduced by a factor 10^6) as well as execution times are given in Tab. 6. These results call for several comments :

- for this particular steady flow test case, the influence of the local solution strategy on the non-linear convergence is rather weak. The reference result is given by the convergence of the global solution strategy based on the Jacobi relaxation method (98 iterations independently of the number of subdomains; for information, increasing the linear threshold from $\varepsilon_g = 10^{-1}$ to $\varepsilon_g = 10^{-2}$ did not result in the reduction of the number of pseudo-time steps to reach the steady state). In comparison, the DDM solver based on the 1 V-cycle(4,4) local solution strategy always yield lower number of pseudo-time steps with a slight degradation when increasing the number of subdomains;
- replacing exact subdomain solutions by approximate solutions using 1 V-cycle(4,4) makes the domain decomposition solver competitive with the global solution strategy. Indeed, we note a 18% reduction of the total execution time for $N_p = 12$ between the global solution strategy and the DDM one. As expected, the DDM solver demonstrates higher parallel efficiencies due to reduced communication overheads.

6.5.2 Transonic flow test case

Steady iso-Mach lines for this test case are visualized on Fig. 19. The results presented below are all characterized by the following points :

- calculations are performed using mesh N3;

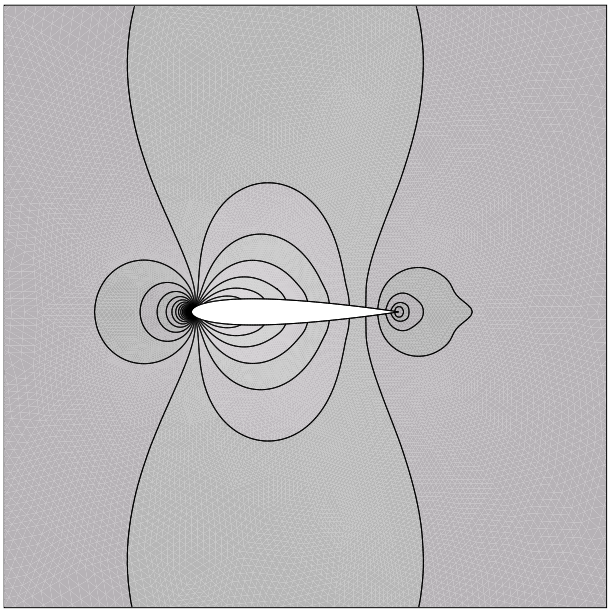


Figure 17: Steady Mach lines for the subsonic flow around the `NACA0012` airfoil

Table 6: Subsonic flow test case : timings for the steady state solution
Global solution strategy (Jacobi, $\varepsilon_g = 10^{-1}$) versus DDM strategy (full GMRES, $\varepsilon_i = 10^{-1}$)

Method	N_p	# it	CPU	Elapsed	% CPU
Jacobi (global system)	4	98	2914 sec	2995 sec	97.2
	8	98	1992 sec	2252 sec	88.5
	12	98	1071 sec	1207 sec	88.9
GMRES/4 V-cycle(2,2)	8	102	3870 sec	4144 sec	93.4
GMRES/1 V-cycle(4,4)	4	93	2686 sec	2748 sec	97.8
	8	94	1728 sec	1802 sec	95.9
	12	96	911 sec	982 sec	92.7

- the CFL number is varying according to the law CFL=$5 \times k_t$ where k_t denotes the time iteration;
- the calculation starts from a uniform flow;
- the local systems induced by the domain decomposition solver are never solved with a high accuracy.

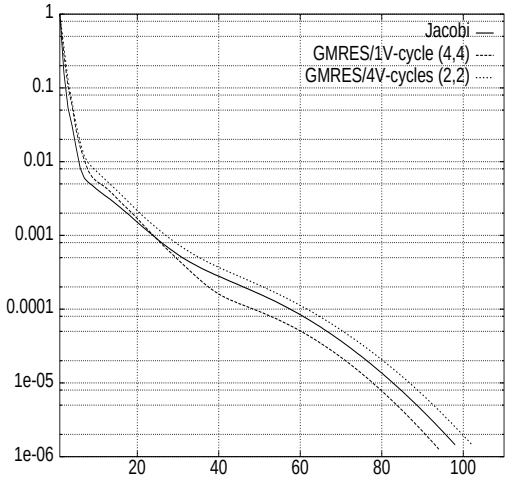


Figure 18: Subsonic flow test case
Non-linear convergence (CFL = 1000, mesh N3, $N_p = 8$)
Global solution strategy (Jacobi, $\varepsilon_g = 10^{-1}$) versus DDM strategy (full GMRES, $\varepsilon_i = 10^{-1}$)

Fig. 20 visualises the non-linear convergence in terms of the normalised energy residual for the following situations :

- the global solution strategy where, at each time step, the linear system (36) is solved using Jacobi relaxations with a linear threshold fixed to $\varepsilon_g = 10^{-1}$ and $\varepsilon_g = 10^{-2}$;
- the proposed DDM strategy where, at each time step, the interface system (42) is solved using full GMRES iterations (without preconditioning) with a linear threshold fixed to $\varepsilon_i = 10^{-1}$ and $\varepsilon_i = 10^{-2}$; moreover, the local linear systems are solved using 1 V-cycle using $\nu_1 = 4$ pre-smoothing and $\nu_2 = 4$ post-smoothing steps (the smoother is a pointwise Gauss-Seidel method) using 3 coarse grid levels (i.e. $N_g = 4$).

Effective number of time steps to convergence (initial normalised energy residual reduced by a factor 10^6) as well as execution times are given in Tab. 7. These results call for several comments :

- it is clear that setting the linear threshold to $\varepsilon_g = 10^{-1}$ in the global solution strategy is not sufficient to guarantee a correct convergence to steady state. As a

matter of fact, the desired level of reduction in the energy residual has not been obtained after 200 pseudo-time steps (this also explain the lower value of the corresponding execution time in Tab. 7). Consequently, it has been necessary to set the linear threshold to the value $\varepsilon_g = 10^{-2}$;

- on the other hand, switching from $\varepsilon_i = 10^{-1}$ to $\varepsilon_i = 10^{-2}$ in the DDM solution strategy did not improve the convergence to steady state. The required number of pseudo-time steps for $\varepsilon_i = 10^{-1}$ is even lower than what is obtained by setting $\varepsilon_g = 10^{-2}$ in the global solution strategy. This suggests that in the latter case, switching to $\varepsilon_g = 10^{-3}$ would have certainly resulted in a further reduction of the required number of pseudo-time steps however at the expense of notably higher execution times;

- by comparing the total execution time of the global solution strategy based on $\varepsilon_g = 10^{-2}$ with that of the DDM solver based on $\varepsilon_i = 10^{-1}$ we conclude that the latter is about 3 times faster than the former.

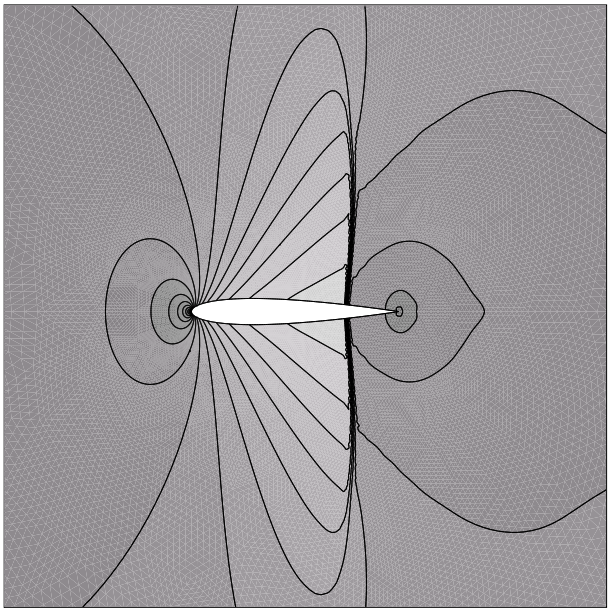


Figure 19: Steady Mach lines for the transonic flow around the `NACA0012` airfoil

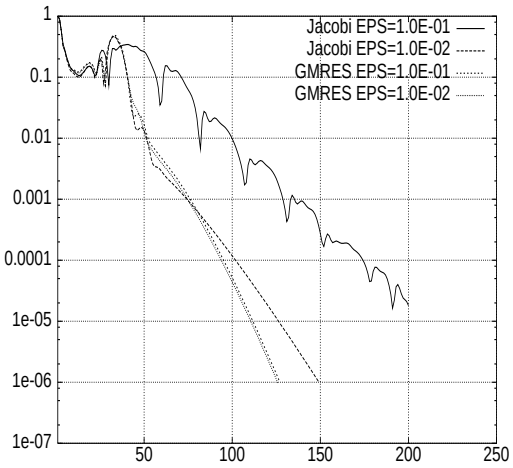


Figure 20: Transonic flow test case
Non-linear convergence (CFL = 1000, mesh N3, $N_p = 8$)
Global solution strategy (Jacobi, $\varepsilon_g = 10^{-1}$ and $\varepsilon_g = 10^{-2}$)
DDM strategy (full GMRES, $\varepsilon_i = 10^{-1}$ and $\varepsilon_i = 10^{-2}$)

Table 7: Transonic flow test case : timings for the steady state solution
Global solution strategy (Jacobi, $\varepsilon_g = 10^{-1}$ and $\varepsilon_g = 10^{-2}$)
DDM strategy (full GMRES, $\varepsilon_i = 10^{-1}$ and $\varepsilon_i = 10^{-2}$)

Method	N_p	# it	CPU	Elapsed	% CPU
Jacobi (global system), $\varepsilon_g = 10^{-1}$	8	200	1825 sec	2017 sec	90.5
Jacobi (global system), $\varepsilon_g = 10^{-2}$	8	149	4524 sec	5095 sec	88.7
GMRES/1 V-cycle(4,4), $\varepsilon_i = 10^{-1}$	8	127	1631 sec	1700 sec	96.0
GMRES/1 V-cycle(4,4), $\varepsilon_i = 10^{-2}$	8	126	2830 sec	2951 sec	95.9

7 Conclusion and future works

In this paper, we have reported on our recent efforts on the design of a non-overlapping domain decomposition algorithm for the solution of two-dimensional compressible inviscid flows. More precisely, we have formulated an additive Schwarz algorithm which is based on interface conditions that are Dirichlet conditions for the characteristic variables corresponding to incoming waves. The algorithm has been introduced in the continuous case for a general linearized and symmetrized hyperbolic system. A concrete implementation has been proposed for the solution of the Euler equations, in the context of a mixed finite element/finite volume solver on unstructured triangular meshes. Within this solver, time integration of the semi-discrete equations is obtained using a linearized backward Euler implicit scheme. Then, the proposed domain decomposition algorithm is used for advancing the solution at each implicit time step. It has also been shown that, through the introduction of interface operators, it is possible to express the proposed domain decomposition algorithm as a Richardson type iteration on interface unknowns. Algebraically speaking, the Schwarz algorithm is equivalent to a Jacobi iteration applied to a linear system whose matrix has a block structure. A substructuring technique can be applied to this matrix in order to obtain a fully implicit scheme in terms of interface unknowns. In our approach, the interface unknowns are numerical fluxes computed using the approximate Riemann solver of Roe[35].

The resulting algorithm has been applied to the computation of a subsonic and a transonic steady flow around a `NACA0012` airfoil. An original aspect of our study consists in the iterative solution of local problems using a multigrid by agglomeration technique. In particular, we have investigated numerically the effect of an approximate solution of local problems on the overall efficiency of the domain decomposed solver. For steady (Euler) flow computations, such a strategy is mandatory to make the domain decomposed solver competitive with classical (global) solution techniques. From this point of view, the proposed domain decomposed solver can also be viewed as a particular form of additive multigrid in which multigrid acceleration is applied on a subdomain basis, these local calculations being coordinated by an appropriate DDM solver for the interface unknowns. The successful application of this strategy is clearly illustrated on Fig. 20 and Tab. 7.

Ongoing efforts and future works concern the following aspects :

- convergence analysis of the additive Schwarz algorithm. Our approach to the evaluation of the convergence rate of the additive Schwarz algorithm (17) relies on a Fourier analysis of the linearized two-dimensional Euler equations in the

context of a stripwise decomposition of a rectangular domain. Similar approaches have been adopted in [28], [29] and [22];

- construction of preconditioners for the interface system (42). We are currently investigating two approaches : on one hand, we study algebraic preconditioning techniques for obtaining approximate inverses to the matrix S (also, we investigate the use of the agglomeration strategy to the construction of a coarse space preconditioner); on the other hand, we study the construction of a preconditioner at the continuous level following the approach proposed in [1];

- extension of the proposed DDM algorithm to the solution of the Navier-Stokes equations for compressible flows. In that case, the formulation of a non-overlapping domain decomposition algorithm requires the definition of combined convective/diffusive flux interface conditions[31]. The main difficulty that we face is the implicit treatment of such interface conditions in the context of the exisiting flow solver which is based on a mixed finite volume /finite element formulation (upwind schemes for the discretisation of convective fluxes/Galerkin approximation of the diffusive fluxes).

Acknowledgements : the authors wish to thank Alain Dervieux and Jean-Antoine Désidéri for their comments and ideas on this work. The first author also acknowledges support from CNES.

Annex 1

We detail here the expressions of the various terms of the matrix of system (41). For simplicity, we consider a two-sudomain decomposition. Then this matrix has the form :

$$\mathcal{M} = \begin{pmatrix} \mathcal{M}_1 & 0 & \mathcal{M}_{12} \\ 0 & \mathcal{M}_2 & \mathcal{M}_{21} \\ \mathcal{F}_1 & \mathcal{F}_2 & \mathrm{I}d \end{pmatrix}$$

where we recall that $\mathcal{M}_1$ (respectively $\mathcal{M}_2$) is the matrix that couples the unknowns associated with vertices internal to Ω_1 (respectively Ω_2) whereas $\mathcal{F}_1$, $\mathcal{F}_2$, $\mathcal{M}_{12}$ and $\mathcal{M}_{21}$ are coupling matrices between internal and interface unknowns.

The implicit scheme is based on the linearization of the first order convective flux (25). For an edge $[s_i, s_j]$ the implicit version of the associated numerical flux is formally written as :

$$\Phi_{ij}^{n+1} = \Phi_{\mathcal{F}}(W_i^n, W_j^n, W_i^{n+1}, W_j^{n+1}, \vec{\nu}_{ij})$$

Noting $U = W_i^n$, $V = W_j^n$, $W = W_i^{n+1}$ and $Z = W_j^{n+1}$ and using a first order Taylor expansion of the implicit flux we obtain :

$$\Phi_{\mathcal{F}}(U, V, W, Z, \vec{\nu}_{ij}) = \Phi_{\mathcal{F}}(U, V, \vec{\nu}_{ij}) + \left(\frac{\partial \Phi}{\partial U}\right)(W - U) + \left(\frac{\partial \Phi}{\partial V}\right)(Z - V) \qquad (50)$$

where :

$$\Phi_{\mathcal{F}}(U, V, \vec{\nu}_{ij}) = \Phi_{\mathcal{F}}(W_i^n, W_j^n, \vec{\nu}_{ij})$$

is the explicit flux. Expression (50) can be simplified in the case where the numerical flux function takes the form :

$$\Phi_{\mathcal{F}}(U, V, \vec{\nu}_{ij}) = H_1(U, V, \vec{\nu}_{ij})U + H_2(U, V, \vec{\nu}_{ij})V$$

For instance, for the numerical flux function associated to the approximate Riemann solver of Roe[35] we have :

$$\begin{aligned} \Phi_{\mathcal{F}}(W_i, W_j, \vec{\nu}_{ij}) &= \mathcal{F}(W_i, \vec{\nu}_{ij}) + \mathcal{A}_R^-(W_i, W_j, \vec{\nu}_{ij})(W_j - W_i) \\ &= \mathcal{A}(W_i, \vec{\nu}_{ij})W_i + \mathcal{A}_R^-(W_i, W_j, \vec{\nu}_{ij})(W_j - W_i) \end{aligned}$$

therefore :

$$\left\{ \begin{array}{lcl} H_1(U,V,\vec{\nu}_{ij}) & = & \mathcal{A}(U,\vec{\nu}_{ij}) - \mathcal{A}_R^-(U,V,\vec{\nu}_{ij}) \\ H_2(U,V,\vec{\nu}_{ij}) & = & \mathcal{A}_R^-(U,V,\vec{\nu}_{ij}) \end{array} \right.$$

In other words, we obtain an approximate linearization (an thus an approximate Jacobian matrix) if we assume :

$$\frac{\partial \Phi}{\partial U} \approx H_1(U,V,\vec{\nu}_{ij}) \text{ and } \frac{\partial \Phi}{\partial V} \approx H_2(U,V,\vec{\nu}_{ij})$$

Note that for the Steger and Warming[39] numerical flux function (see Eq. (31)) we have :

$$\left\{ \begin{array}{lcl} H_1(U,V,\vec{\nu}_{ij}) & = & \mathcal{A}^+(U,\vec{\nu}_{ij}) \\ H_2(U,V,\vec{\nu}_{ij}) & = & \mathcal{A}^-(V,\vec{\nu}_{ij}) \end{array} \right.$$

We can now write the nodal equations associated to the matrix-vector product :

$$\mathcal{M}\,(\delta W_1\ ,\ \delta W_2\ ,\ \Phi)^T = (b_1\ ,\ b_2\ ,\ 0)^T$$

where $\delta W_{1,2} = W_{1,2}^{n+1} - W_{1,2}^n$. For simplicity we do not take into account the terms associated to the convective fluxes at physical boundaries (i.e. on Γ_b and Γ_∞). Let $e_{ij}^I = [s_i^I, s_j^I]$ denote an interface edge (see Fig. 3). We also denote by $N(i)$ the set of neighboring vertices of s_i that are purely internal to a subdomain (i.e. they are the end-points of an internal edge) and by $N^I(i)$ the set of neighboring vertices of s_i that are end-points of an interface edge (i.e. s_j is an internal vertex of the neighboring subdomain).

Internal unknowns : these are components of δW_1 and δW_2. For a vertex s_i internal either to Ω_1 or Ω_2 we obtain :

$$(D_{1,2})_i^n (\delta W_{1,2})_i + \sum_{j \in N(i)} (E_{1,2})_{ij}^n (\delta W_{1,2})_j + \sum_{j \in N^I(i)} (\mathcal{M}_{12,21})_{i,e_{ij}^I}^n (\delta \Phi)_{e_{ij}^I} = (b_{1,2})_i^n \qquad (51)$$

For the two-dimensional Euler equations, $(D_{1,2})_i^n$, $(\mathcal{M}_{12,21})_{i,e_{ij}^I}^n$ and $(E_{1,2})_{ij}^n$ are 4×4 matrices :

$$\left\{\begin{array}{lcl} (D_{1,2})_i^n & = & \dfrac{\text{area}(C_i)}{\Delta t^n} + \displaystyle\sum_{j\in N(i)} \left[\mathcal{A}(W_i^n, \vec{\nu}_{ij}) - \mathcal{A}_R^-(W_i^n, W_j^n, \vec{\nu}_{ij})\right] \\ (E_{1,2})_{ij}^n & = & \mathcal{A}_R^-(W_i^n, W_j^n, \vec{\nu}_{ij}) \\ (\mathcal{M}_{12})_{i,e_{ij}^I}^n & = & P^-(\tilde{W}^n) \\ (\mathcal{M}_{21})_{i,e_{ij}^I}^n & = & -\ P^+(\tilde{W}^n) \\ (b_{1,2})_i^n & = & -\displaystyle\sum_{j\in N(i)} \Phi_{\mathcal{F}}(W_i^n, W_j^n, \vec{\nu}_{ij}) \end{array}\right. \tag{52}$$

where $P^-(\tilde{W}^n)$ and $P^+(\tilde{W}^n)$ are given by Eq. (39).

Interface unknowns : these are components of $\delta\Phi$, each of them being associated to an interface edge (see Fig. 3). Assume that each interface edge $e_{ij}^I = [s_i^I, s_j^I]$ is such that s_i^I is an internal vertex of Ω_1 and s_j^I is an internal vertex of Ω_2. Then, for each e_{ij}^I we have :

$$(\delta\Phi)_{e_{ij}^I} + (\mathcal{F}_1)_{e_{ij}^I,i}^n (\delta W_1)_i + (\mathcal{F}_2)_{e_{ij}^I,j}^n (\delta W_2)_j = 0 \tag{53}$$

with :

$$\left\{\begin{array}{lcl} (\mathcal{F}_1)_{e_{ij}^I,i}^n & = & \mathcal{A}_R^-(W_i^n, W_j^n, \vec{\nu}_{ij}) \\ (\mathcal{F}_2)_{e_{ij}^I,j}^n & = & -\ \mathcal{A}_R^+(W_i^n, W_j^n, \vec{\nu}_{ij}) \end{array}\right. \tag{54}$$

References

[1] Y. Achdou, P. Le Tallec, Nataf F., and M. Vidrascu. A domain decomposition preconditionner for an advection-diffusion problem. *Comp. Meth. in Appl. Mech. and Eng.*, 1998. to appear.

[2] T. Barth. *Numerical methods for gasdynamics systems on unstructured meshes*, volume 5 of *Lecture Notes in Computatational Science and Engineering*, pages 195–285. Springer Verlag, 1999.

[3] X.-C. Cai and O.B. Widlund. Domain decomposition algorithms for indefinite elliptic problems. *SIAM J. Sci. Stat. Comput.*, 13:243–259, 1992.

[4] C. Carlenzoli and A. Quarteroni. Adaptive domain decomposition methods for advection-diffusion problems. In Babuska *et al.*, editor, *Modeling, mesh generation, and adaptive numerical methods for partial differential equations*, volume 75 of *IMA Volumes in Mathematics and its Applications*, pages 169–199. Springer Verlag, 1995.

[5] G. Carré. An implicit multigrid method by agglomeration applied to turbulent flows. *Computers & Fluids*, (26):299–320, 1997.

[6] S. Clerc. *Etude de schémas décentrés implicites pour le calcul numérique en mécanique des fluides. Résolution par décomposition de domaine.* PhD thesis, Université de Paris VI, 1997.

[7] J.-A. Désidéri. *Modèles discrets et schémas itératifs. Application aux algorithmes multigrilles et multidomaines.* Hermes, 1998.

[8] F. d'Hennezel, P. Le Tallec, and M. Vidrascu. A parallel algorithm for advection-diffusion problem using domain decomposition. Technical Report 33, STPA-ONERA, 1992.

[9] B. Engquist and A. Majda. Absorbing boundary conditions for the numerical simulation of waves. *Math. Comp.*, 31:629–651, 1977.

[10] C. Farhat and S. Lanteri. Simulation of compressible viscous flows on a variety of mpps : computational algorithms for unstructured dynamic meshes and performance results. *Comp. Meth. in Appl. Mech. and Eng.*, 119:35–60, 1994.

[11] L. Fezoui and A. Dervieux. Finite element non-oscillatory schemes for compressible flows. In *Eighth France-U.S.S.R.-Italy Joint Symposium on Computational Mathematics and Applications*, number 370 in IAN, Pavie, Italie, 1989.

[12] L. Fezoui and B. Stoufflet. A class of implicit upwind schemes for Euler simulations with unstructured meshes. *J. of Comp. Phys.*, 84:174–206, 1989.

[13] F. Gastaldi and L. Gastaldi. On a domain decomposition for the transport equation : theory and finite element approximation. *IMA J. Numer. Anal.*, 14:111–135, 1993.

[14] F. Gastaldi, L. Gastaldi, and A. Quarteroni. Adaptive domain decomposition methods for advection-dominated equations. *East-West J. Numer. Math.*, 4:165–206, 1996.

[15] F. Gastaldi, L. Gastaldi, and A. Quarteroni. Adn and arn domain decomposition methods for advection-diffusion equations. In D. Keyes P. Bjorstad, M. Espedal, editor, *Proceedings of the 9th International Conference on Domain Decompositon Methods in Science and Engineering*. Wiley & Sons, 1997.

[16] E. Godlewski and P.-A. Raviart. *Numerical approximation of hyperbolic systems of conservation laws*, volume 118 of *Applied Mathematical Sciences*. Springer Verlag, 1996.

[17] W. Hackbusch. *Multigrid methods and applications*, volume 4 of *Springer series in Computationl Mathematics*. Springer Verlag, 1985.

[18] L. Halpern. Artificial boundary conditions for the advection-diffusion equation. *Math. Comp.*, 174:425–438, 1986.

[19] L. Halpern. Conditions aux limites artificielles pour un système incomplètement parabolique. *C. R. Acad. Sci. Paris*, 307:413–416, 1988.

[20] L. Halpern and M. Schatzman. Artificial boundary conditions for incompressible viscous flows. *SIAM J. Matrix Anal.*, 20:308–353, 1989.

[21] P.-W Hemker. On the order of prolongations and restrictions in multigrid procedures. *J. Comput. Appl. Math.*, (32):423–429, 1990.

[22] C. Japhet. *Méthode optimisée d'ordre 2. Application à l'équation d'advection-diffusion*. PhD thesis, Université Paris XIII, 1998.

[23] M.-H. Lallemand, Steve S., and Dervieux A. Unstructured multigridding by volume agglomeration : current status. *Computers & Fluids*, 21:397–433, 1992.

[24] S. Lanteri. Parallel solutions of compressible flows using overlapping and non-overlapping mesh partitioning strategies. *Parallel Computing*, 22:943–968, 1996.

[25] D. J. Mavriplis. Directional agglomeration multigrid techniques for hight-reynolds number viscous flows. Technical Report 98-7, ICASE, January 1998.

[26] D. J. Mavriplis and V. Venkatakrishnan. Agglomeration multigrid fot two-dimensional viscous flows. *J. Comput. Phys.*, 24:553–570, 1995.

[27] D. J. Mavriplis and V. Venkatakrishnan. A 3d agglomeration multigrid solver for the reynolds-average navier-stokes equations on unstructure meshes. *Int. J. Numer. Meth. Fluids*, 23:527–544, 1996.

[28] F. Nataf. On the use of open boundary conditions in block gauss-seidel methods for the convection-diffusion equation. Technical Report RI284, Centre de Mathématiques Appliquées, Ecole Polytechnique, 1993.

[29] F. Nataf and F. Nier. Convergence rate of some domain decomposition methods for overlapping and nonoverlapping subdomains. Technical Report RI306, Centre de Mathématiques Appliquées, Ecole Polytechnique, 1996.

[30] F. Nataf, F. Rogier, and E. de Sturler. Optimal interface conditions for domain decomposition methods. Technical Report RI301, Centre de Mathématiques Appliquées, Ecole Polytechnique, 1994.

[31] A. Quarteroni and L. Stolcis. Homogeneous and heterogeneous domain decomposition methods for compressible flow at high reynolds numbers. Technical Report 33, CRS4, 1996.

[32] A. Quarteroni and A. Valli. *Theory and application of Steklov-Poincaré operators for boundary value problems*, pages 179–203. Kluwer Academic Publishers, 1991.

[33] A. Quarteroni and A. Valli. Domain decomposition methods for compressible flows. Technical report, EPFL, 1999.

[34] A. Quarteroni and A. Valli. *Domain Decomposition Methods for Partial Differential Equations*. Oxford University Press, 1999.

[35] P.L. Roe. Approximate Riemann solvers, parameter vectors and difference schemes. *J. of Comp. Phys.*, 43:357–371, 1981.

[36] Y. Saad and H. Schultz. Gmres : Generalized minimal residual algorithm for solving non-symmetric linear systems. *SIAM J. Sci. Stat. Comput.*, 7:856–869, 1986.

[37] B. Smith. An optimal domain decomposition preconditionner for the finite element solution of linear elasticity. *SIAM J. Sci. Stat. Comput.*, 13:364–379, 1992.

[38] B. Smith, P. Bjorstad, and W. Gropp. *Domain Decomposition and Parallel Multilevel Methods for Elliptic Partial Differential Equations.* Cambridge University Press, 1996.

[39] J. Steger and R. F. Warming. Flux vector splitting for the inviscid gas dynamic with applications to finite difference methods. *J. of Comp. Phys.*, 40:263–293, 1981.

[40] B. Van Leer. Towards the ultimate conservative difference scheme V : a second-order sequel to Godunov's method. *J. of Comp. Phys.*, 32:361–370, 1979.

[41] P. Wesseling. *An introduction to multigrid methods.* John Wiley & Sns, 1991.

[42] J. Xu and X.-C. Cai. A preconditionned gmres method for nonsymmetric or indefinite problems. *Math. Comp.*, 59:311–319, 1992.

Unité de recherche INRIA Sophia Antipolis
2004, route des Lucioles - B.P. 93 - 06902 Sophia Antipolis Cedex (France)

Unité de recherche INRIA Lorraine : Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - B.P. 101 - 54602 Villers lès Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38330 Montbonnot St Martin (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - B.P. 105 - 78153 Le Chesnay Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, B.P. 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399